SIGPwny

Embedded FA2025 • 2025-10-27

Side-Channel Attacks

Jake and Minh

Overview

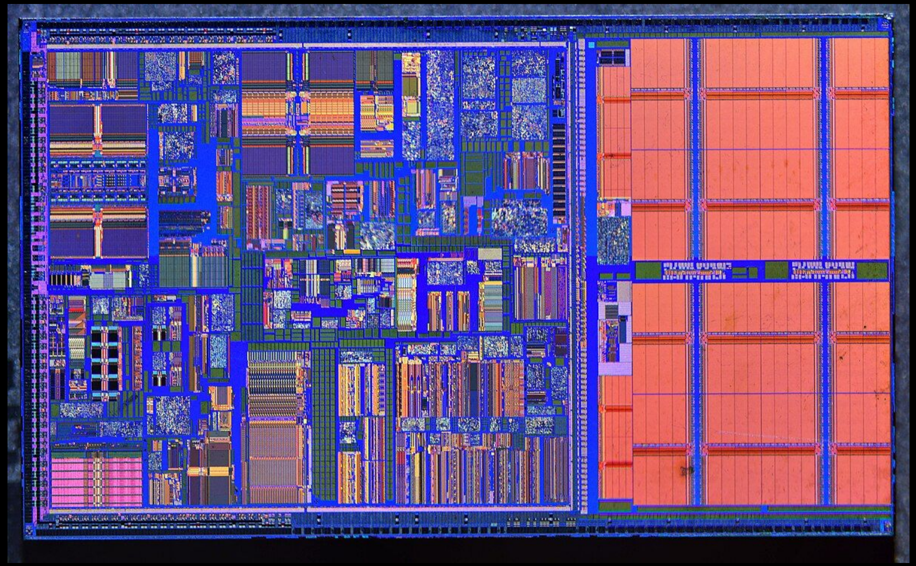
- What is Side-Channels Analysis (SCA)?
- Timing Side-Channels
- Power analysis
  - Simple Power Analysis (SPA)
  - Differential Power Analysis (DPA)
  - Correlation Power Analysis (CPA)
- Electro-magnetic

Side-Channel Analysis

- Side-Channel: Indirect source of information about a system
- SCA: Leaking private information by side-channels
- Some side-channels:
  - Timing
  - Power use
  - EM/RF emissions
- General purpose computing:
  - Meltdown/Spectre target cache side-channels
  - Timing analysis against naïve crypto implementations
- Embedded devices may be exposed to invasive physical threats

Side-Channel Analysis

- When a processor is executing instructions, it draws power
- This power draw is related to the operations it is performing
- Further, the power draw is related to the operands (i.e. values) being manipulated
- Additionally, there will be data-dependent EM/RF emissions



Power Side-Channels

RSA Primer

- Alice wants to send a message to Bob that only Bob can decrypt
- Bob distributes a public key (e, n) and retains a secret key d
- Alice pads her secret message to produce m such that $m < n$
- Alice produces the ciphertext $c \equiv m^e \pmod{n}$
- Bob can recover the message $c^d \equiv (m^e)^d \equiv m \pmod{n}$
- Observe that decryption involves using the secret key d as an exponent

Exponentiation by Squaring

- A simple algorithm to compute `r = x`^{n} is as follows:

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

$r \leftarrow 1$

$x \leftarrow 2$

$n \leftarrow 5$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
   if n is odd:
      r = r * x
   x = x * x
   n = n // 2
```

$r \leftarrow 1$

$x \leftarrow 2$

$n \leftarrow 5$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

$r \leftarrow 1$

$x \leftarrow 2$

$n \leftarrow 5$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x (2)
    x = x * x
    n = n // 2
```

$r \leftarrow 1$

$x \leftarrow 2$

$n \leftarrow 5$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
   if n is odd:
      r = r * x (2)
   x = x * x
   n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 2$

$n \leftarrow 5$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
   if n is odd:
      r = r * x
   x = x * x (4)
   n = n // 2
```

r ← 2

x ← 2

n ← 5

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x (4)
    n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 4$

$n \leftarrow 5$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2 (2)
```

$r \leftarrow 2$

$x \leftarrow 4$

$n \leftarrow 5$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
   if n is odd:
      r = r * x
   x = x * x
   n = n // 2 (2)
```

$r \leftarrow 2$

$x \leftarrow 4$

$n \leftarrow 2$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 4$

$n \leftarrow 2$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 4$

$n \leftarrow 2$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
   if n is odd:
      r = r * x
   x = x * x (16)
   n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 4$

$n \leftarrow 2$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x (16)
    n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 16$

$n \leftarrow 2$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2 (1)
```

$r \leftarrow 2$

$x \leftarrow 16$

$n \leftarrow 2$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2 (1)
```

$r \leftarrow 2$

$x \leftarrow 16$

$n \leftarrow 1$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 16$

$n \leftarrow 1$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 16$

$n \leftarrow 1$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x (32)
    x = x * x
    n = n // 2
```

$r \leftarrow 2$

$x \leftarrow 16$

$n \leftarrow 1$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x (32)
    x = x * x
    n = n // 2
```

$r \leftarrow 32$

$x \leftarrow 16$

$n \leftarrow 1$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x (256)
    n = n // 2
```

$r \leftarrow 32$

$x \leftarrow 16$

$n \leftarrow 1$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x (256)
    n = n // 2
```

$r \leftarrow 32$

$x \leftarrow 256$

$n \leftarrow 1$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
   if n is odd:
      r = r * x
   x = x * x
   n = n // 2 (0)
```

r ← 32

x ← 256

n ← 1

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2 (0)
```

r ← 32

x ← 256

n ← 0

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

$r \leftarrow 32$

$x \leftarrow 256$

$n \leftarrow 0$

Exponentiation by Squaring

- Computing 2^5

```
r = 1
while n > 0:
    if n is odd:
        r = r * x
    x = x * x
    n = n // 2
```

- We have $r = 32 = 2^5$, as desired

r ← 32

x ← 256

n ← 0

Exponentiation by Squaring

- The algorithm is correct, but it has a serious issue
- Recall that the decryption key `d` is Bob's exponent
- The algorithm performs an extra step for each `1` bit in the exponent:

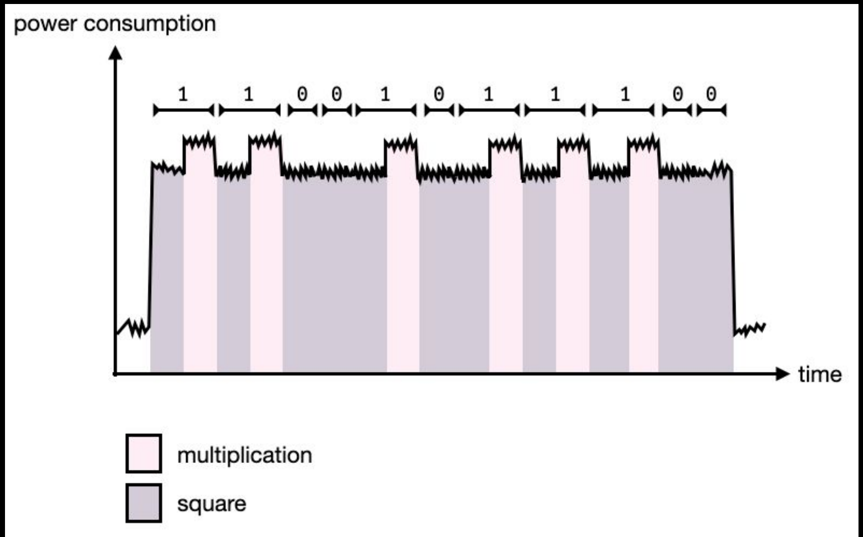
```
if n is even:
    r = r * x
```

- If we could watch the execution path over time, we could see if each bit of the key is a `0` or `1`

Simple Power Analysis

- Well, different instructions may have different power characteristics
- Simple Power Analysis (SPA) refers to visually inspecting a power trace to leak secrets
- For exponentiation by squaring, the `n is even` condition corresponds with an additional multiply step

```
while n > 0:
    if n is even:
        r = r * x (multiply)
    x = x * x (square)
    n = n // 2
```



Note: the square and multiply steps are reordered in this example

Simple Power Analysis

- SPA is great when your power traces have obvious features
- However, hardening against these obvious features isn't very difficult
- A constant-time AES implementation is unlikely to reveal key material visibly in the power trace

Advanced Power Analysis

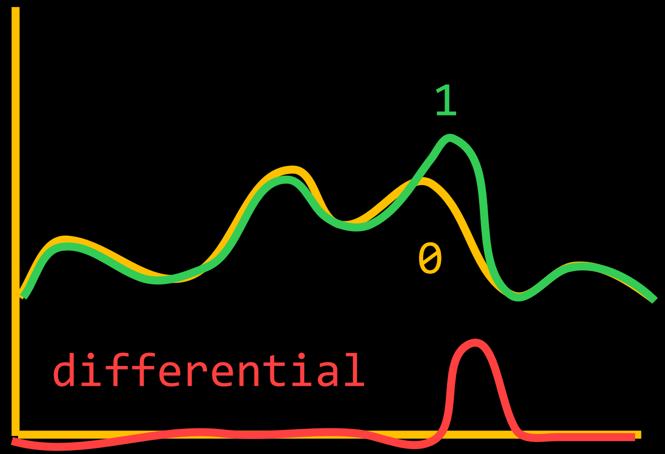
- Instead, we can gather many power traces of encryption/decryption over different inputs
- Each trace on its own isn't useful
- However, let's assume that the device's power consumption depends on the data being processed
  - hamming weight model (more 1 bits, more power)
  - hamming distance (more bits flipped, more power)
- It then follows that some function of the key material and input data is correlated with power consumption

Advanced Power Analysis

- For example, let's say encryption takes your input byte `a` and a secret byte `b` to compute some intermediate value `c`
  - i.e. `c = a ^ b`
- Hamming weight model would suggest that more bits being set in `c` would draw more power
  - e.g. `c = 0xff` is heavier than `c = 0x00` so it draws more power
- Therefore, when all bits of `a` and `b` match, we would have the lowest power draw in setting `c`

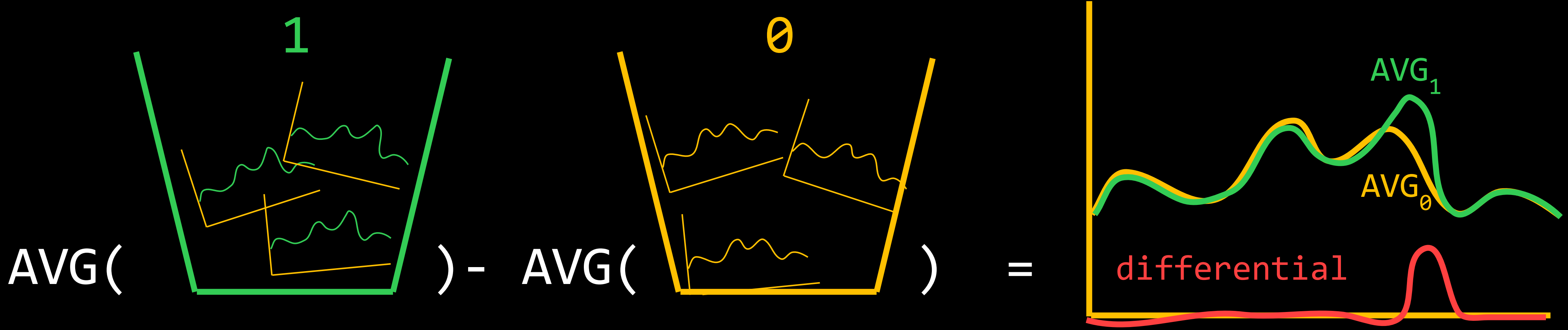
Differential Power Analysis

- Based on our understanding of how processors consume power when performing operations, each bit involved in a computation should contribute slightly to the power trace
- Therefore, if we have two idealized (non-noisy and aligned) power traces only differing in a single bit, we would expect a spike at some point in the differential
  - i.e. the power traces are equal except at one point due that bit differing
- Averaging many traces with uncorrelated noise in other positions will approximate this ideal



Differential Power Analysis

- Differential Power Analysis (Kocher, et al.) is the first work to take advantage of these observations
- It focuses on recovering an intermediate DES key
- Hypothesizes that each value for the key may be taken
- Using the key guess and the ciphertext, we compute some intermediate value that influences the power trace
- Separate the traces into buckets for each bit of the intermediate



Correlation Power Analysis

- Benefitted from the earlier work of DPA and further research
- Introduces a more advanced statistical technique
- Instead of using single bit differences, we can use a correlation coefficient with a leakage model
- Introduces the more advanced hamming distance model to model power draw from bits transitioning from one value to another
- Additionally, the method overcomes some issues observed in DPA
  - DPA assumes wrong guesses give indistinguishable buckets (resulting in false detections)
  - DPA often requires many more samples to converge

Power Analysis Recap

Simple Power Analysis

- Use when you have clear features in your power trace

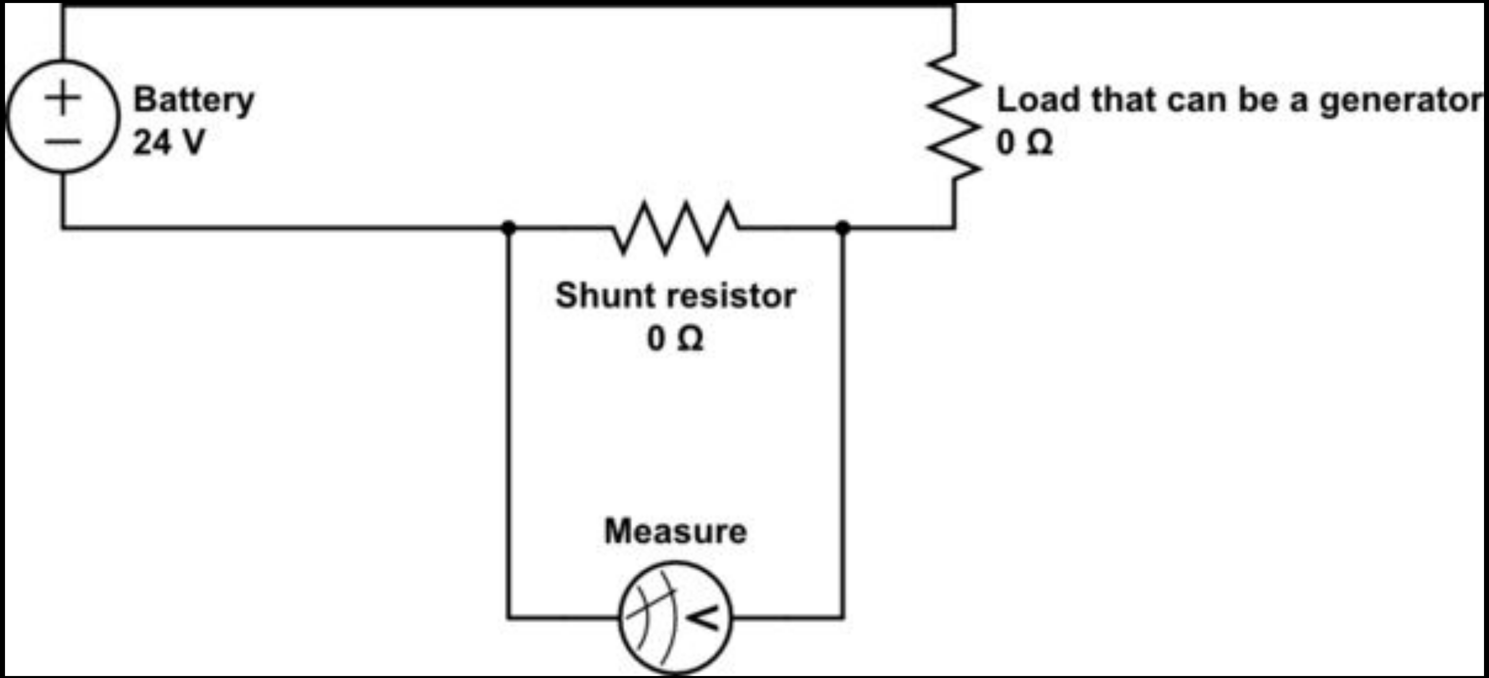
Differential Power Analysis

- First sophisticated statistical attack on power traces
- Makes guesses for key values and buckets traces accordingly

Correlation Power Analysis

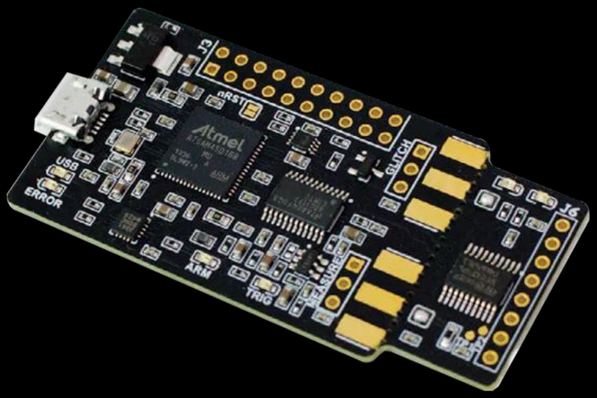
- More useful in practice
- Uses a more advanced statistical method to overcome limitations of DPA

How to perform SCA?

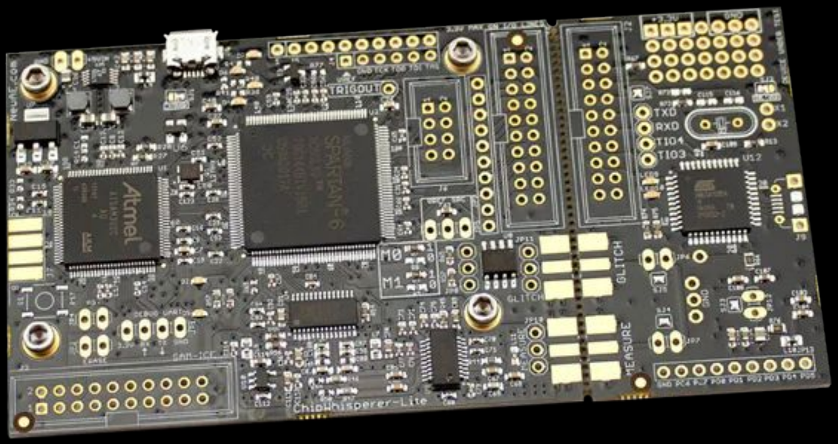

Battery
24 V
Load that can be a generator
0 Ω
Shunt resistor
0 Ω
Measure

ChipWhisperer (CW)

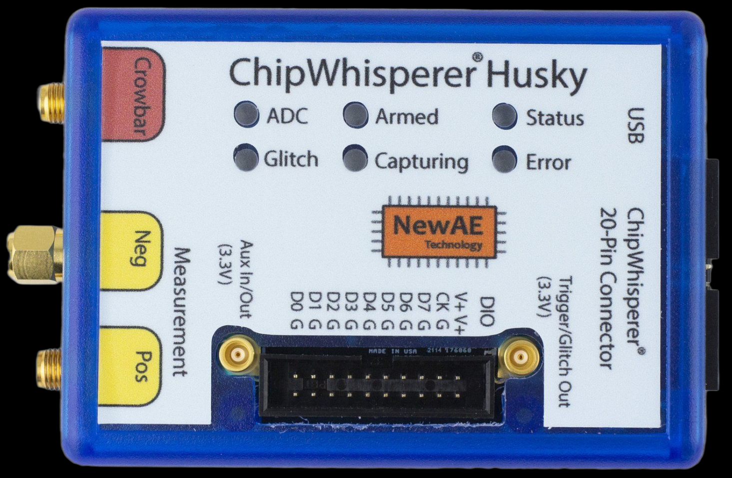
- The ChipWhisperer is a platform for carrying out hardware attacks
  - Anything from side-channel analysis to voltage glitching
- Platform meaning:
  - Attacker hardware
  - Target instrumentation
  - Software library



CW-Nano ($60)



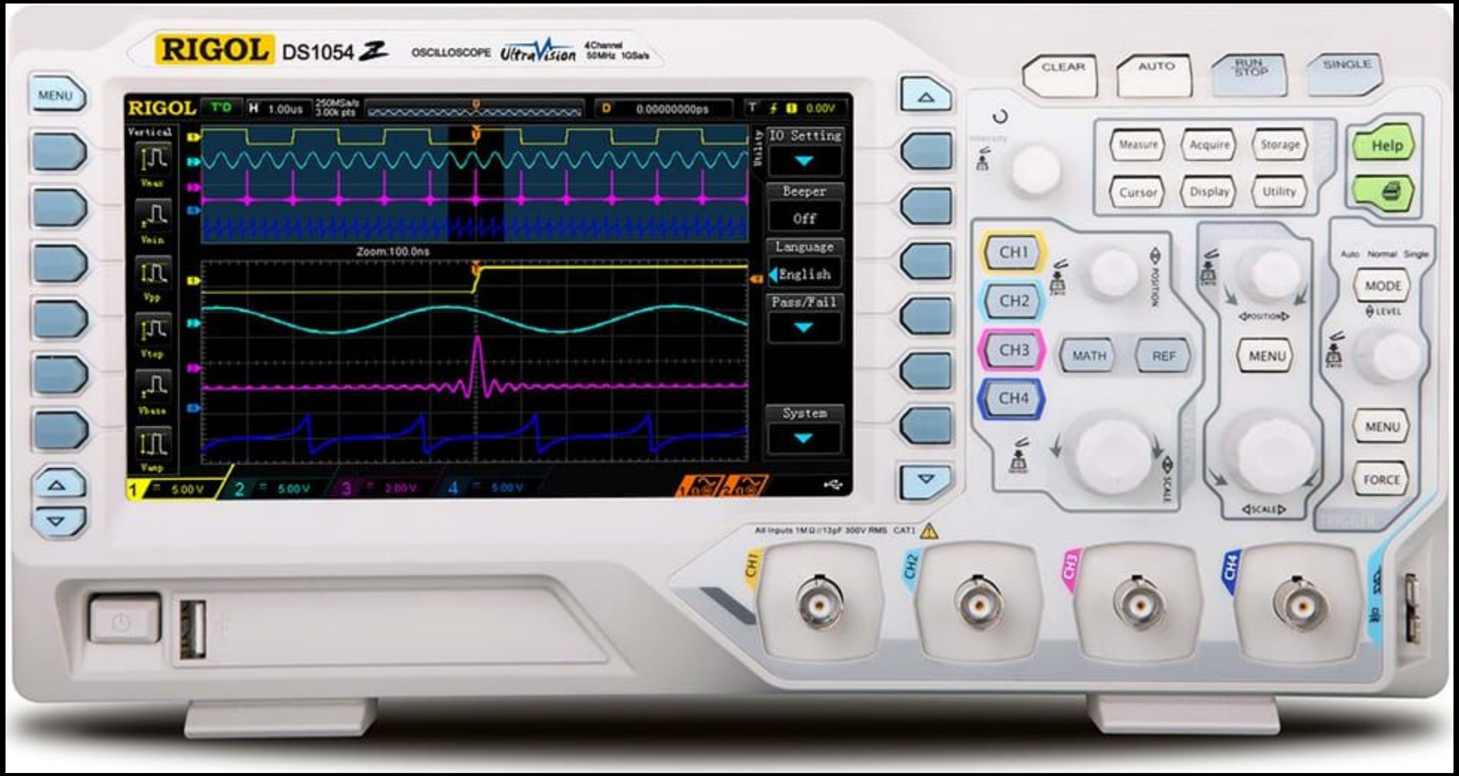
CW-Lite ($370)



CW-Husky ($640)

Oscilloscope

- We can also use an oscilloscope to measure and capture power traces!



Next Meetings

2025-11-03 • Next Monday

- Fault Injection Lab with ChipWhisperer Nano!
- We'll be explaining fault injection as well as letting you use a CW-Nano to perform a voltage glitching attack!

Meeting content can be found at sigpwny.com/meetings.

SIGPwny