

Information Security

System Security 3 - Physical Side-Channel and Fault Attacks

19.11.2021





“Human Side-Channel Analysis”





“If the attacker can execute code
... they have already won”

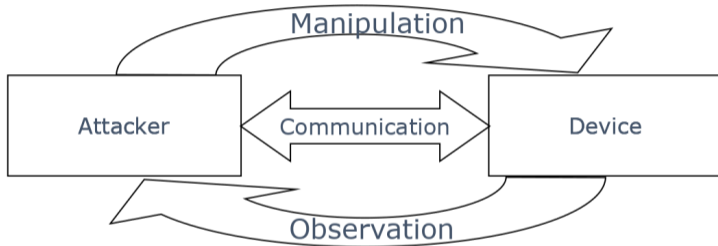
Applications Exposed to Physical Attacks



```
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
```

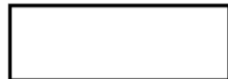


Physical Attack Principle



~~SECRET~~

(b) (3) - P.L. 86-36



Approved for Release by NSA on
09-27-2007, FOIA Case # 51633

TEMPEST: A Signal Problem

**The story of the discovery
of various compromising radiations
from communications and Comsec equipment.**

impractical—hydraulic techniques—to replace the electrical—were tried and abandoned, and experiments were made with different types of batteries and motor generators, in attempts to lick the power-line problem. None was very successful.

During this period, the business of discovering new TEMPEST threats, or refining techniques and instrumentation for detecting, recording, and analyzing these signals, progressed more swiftly than the art of suppressing them. Perhaps the attack is more exciting than the defense—something more glamorous about finding a way to read one of these signals than going through the drudgery necessary to suppress that whacking great spike first seen in 1943. At any rate, when they turned over the next rock, they found the acoustic problem under it. Phenomenon No. 5.

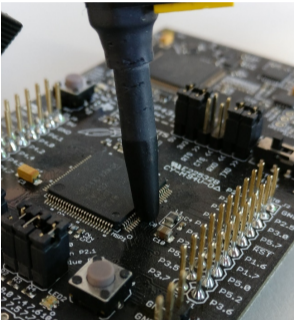
Acoustics

We found that most acoustic emanations are difficult to exploit if the microphonic device is outside of the room containing the source equipment; even a piece of paper inserted between, say, an offending keyboard and a pick-up

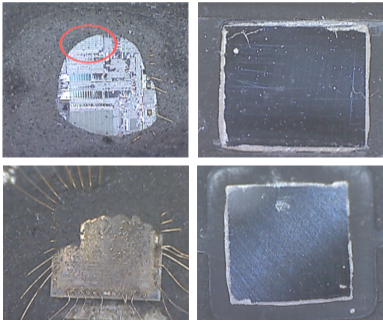


- Behavior of the attacker
 - Side-channel attack: passively observe physical properties
 - Fault attack: actively manipulate device to induce faults
- Degree of invasiveness
 - Non-invasive: Device is not altered physically
 - Semi-invasive: De-packaging, no electrical contact to internal signals
 - Invasive: No limits

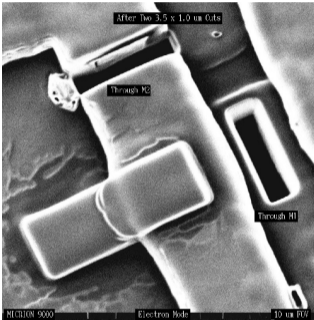
Degree of Invasiveness



Non-Invasive



Semi-Invasive



Invasive

Side-Channel Attacks



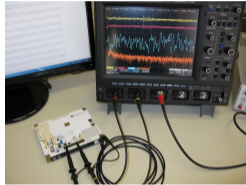
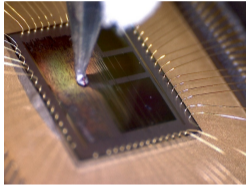
side channel
= obtaining meta-data and
deriving secrets from it

CHANGE MY MIND

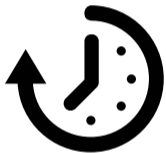


Any computation influences physical properties (meta-data)

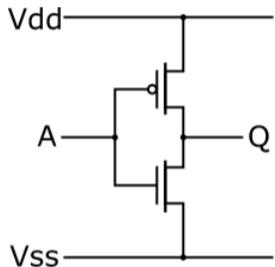
- Computations depend on secrets (data)
- We observe properties (meta-data) to infer secrets (data)



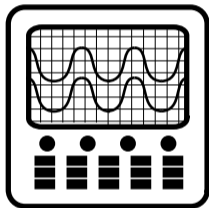
- Timing
- Power consumption
- EM emanations
- Sound
- ...



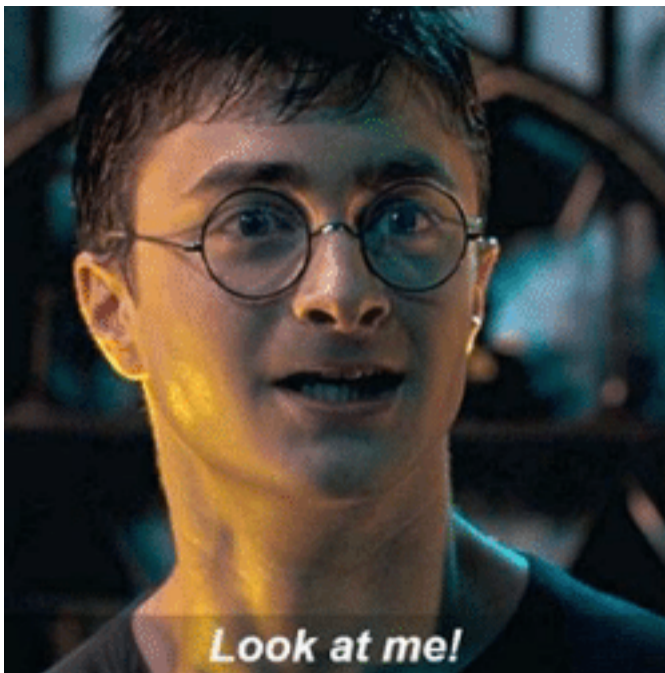
- Often overlooked / ignored
 - “outside of threat model”
 - implementation bugs
 - Sometimes even on certified devices (e.g., Minerva and TPM-Fail)
- Solution: Make everything constant-time?



- Complementary Metal Oxide Semiconductor
- Today's digital circuits
- Nice properties:
 - high noise immunity
 - low power consumption
 - (Only switching draws power)
- Wait a second... switching draws power?



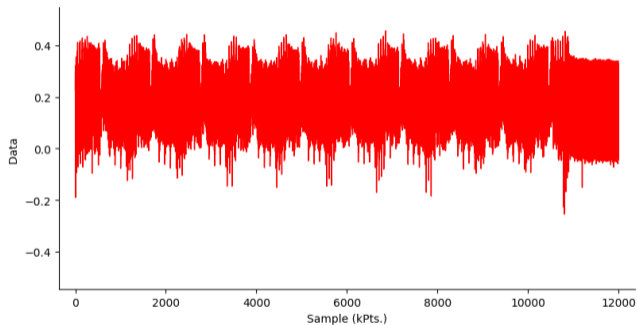
- Different instructions / data → different switching
- Idea: Measure power consumption during operation
 - sampling rate up to gigasamples (10^9 measurements per second)
 - a measured power-consumption curve is called a **power trace**
- First signal-processing step?



Look at me!



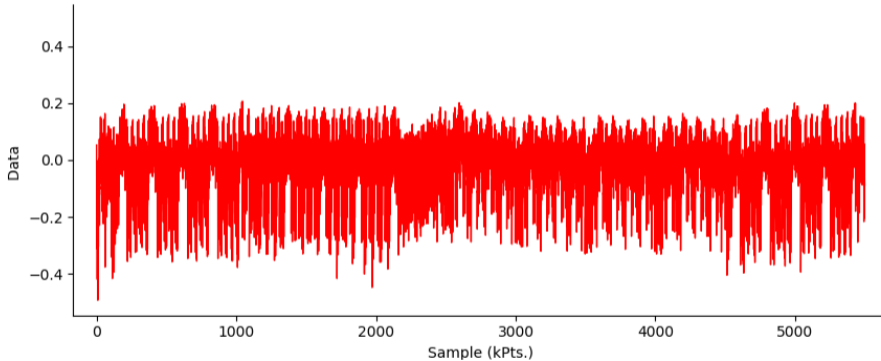
Power Consumption - An Example Trace

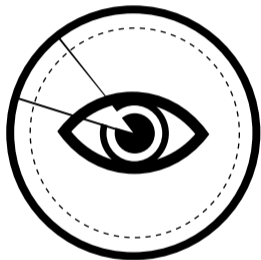


What do we see here?

10 rounds of AES

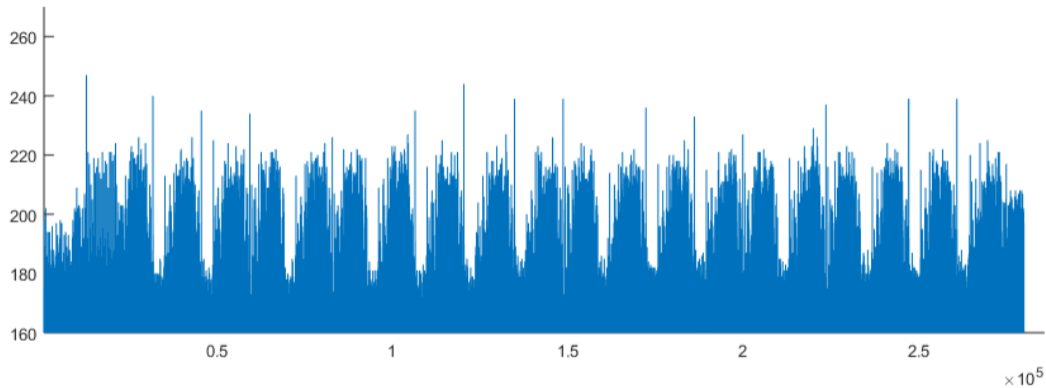
Power Consumption - Zoom in on Single Round



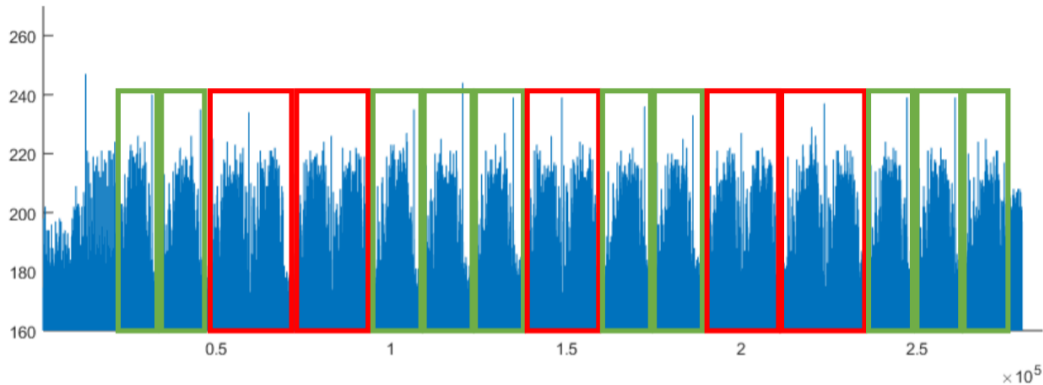


- Operations / Instructions
 - Repeated patterns and variations of patterns
 - Loops, repeated operations, taken branches
 - Learn control flow / instruction sequence
 - Jumps in power consumption profile
 - Memory accesses (especially EEPROM or flash programming)
 - Access to peripherals (e.g., co-processors, I/O)
- Can we exploit that?

Our First Power-Analysis Attack






Our First Power-Analysis Attack



We see a power trace of an RSA exponentiation ($m = c^d \pmod n$)
How to get the key from that?

 RSA decryption: $m = c^d \bmod n$, where n has ≥ 2048 bits

 Efficient implementation?

-  Compute $(c^d) \bmod n$?
-  c and d are also ≥ 2048 bits
-  c^d has more than 2^{2048} bits!

 Some reminders for modular arithmetic:

- $a \cdot b \bmod n = (a \bmod n) \cdot (b \bmod n) \bmod n$
- $c^{a+b} \bmod n = (c^a \bmod n) \cdot (c^b \bmod n) \bmod n$
- $c^{a \cdot b} \bmod n = (c^a \bmod n)^b \bmod n$


- Look at exponent d in binary: $d_i = i$ th bit of d , where $d_0 = \text{LSB}$
- Recursive decomposition of exponentiation:
 - We can write $d = 2 \cdot \lfloor d/2 \rfloor + (d \bmod 2) = 2 \cdot (d \gg 1) + d_0$
In the exponent, we get: $c^d = (c^{\lfloor d/2 \rfloor})^2 \cdot c^{d_0}$
 - But $c^{\lfloor d/2 \rfloor}$ is still way too large, so repeat:
 $c^{\lfloor d/2 \rfloor} = (c^{\lfloor d/4 \rfloor})^2 \cdot c^{\lfloor d/2 \rfloor \bmod 2} = (c^{\lfloor d/4 \rfloor})^2 \cdot c^{d_1}$
 - ... until $\lfloor d/2^x \rfloor = 1 \rightarrow c^{\lfloor d/2^x \rfloor} = c$
- Iterative version: Start at $\lfloor d/2^x \rfloor = 1$ and work our way up

- Algorithm: Left-to-right *Square-and-Multiply* exponentiation

```
 $m \leftarrow 1$  // init  
for  $i = 2047$  downto  $0$  : // scan bits from MSB to LSB  
     $m \leftarrow m^2 \bmod n$  // squaring:  $c^x = (c^{\lfloor x/2 \rfloor})^2 \cdot c^{x_0}$   
    if  $d_i = 1$  then: // if bit is set (else  $x_0 = 0 \rightarrow c^{x_0} = 1$ , can skip mult.)  
         $m \leftarrow m \cdot c \bmod n$  // then multiply:  $c^x = (c^{\lfloor x/2 \rfloor})^2 \cdot c^{x_0}$ 
```

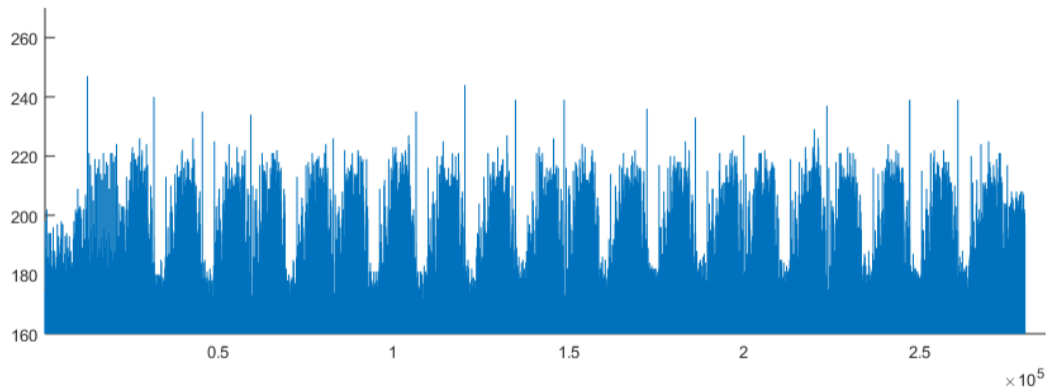
- Example: $d = 26 = 11010_b \rightarrow c^{26} = (((1^2 \cdot c)^2 \cdot c)^2 \cdot c)^2$

Excursion: Efficient Implementation of Modular Exponentiation

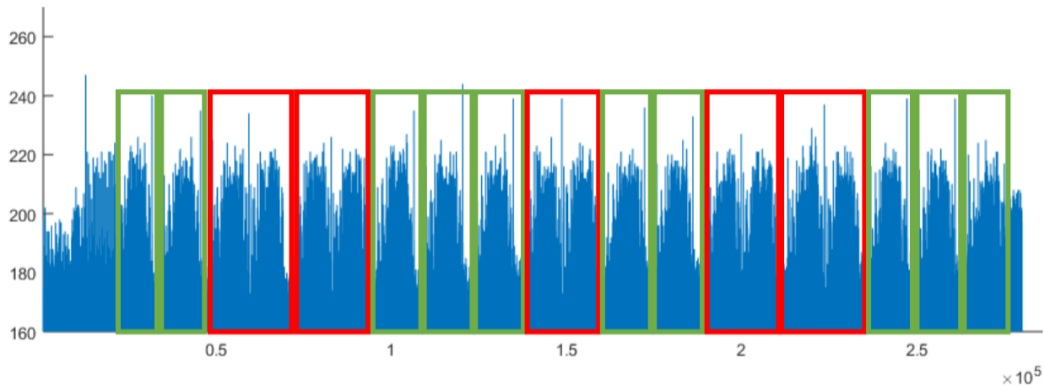
$$M = C^d \pmod n$$


$$\text{Result} = \underbrace{\text{Result} \times \text{Result}}_{\text{square}} \times \underbrace{C}_{\text{multiply}}$$

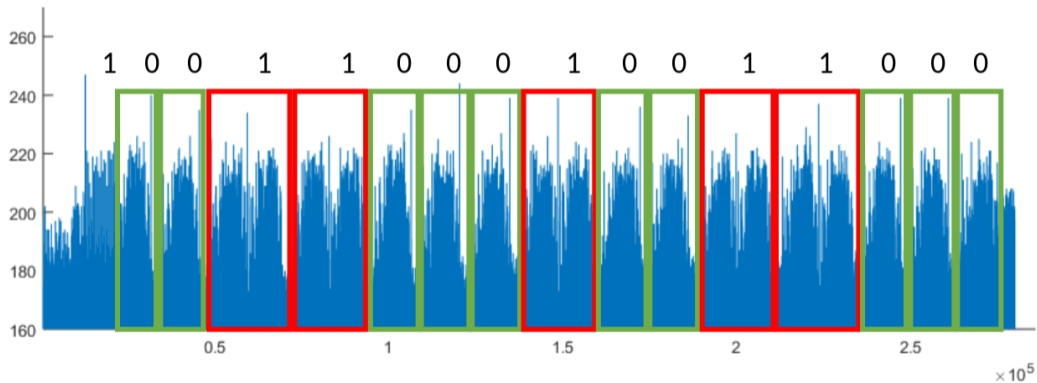
Our First Power-Analysis Attack - Key Recovery



Our First Power-Analysis Attack - Key Recovery



Our First Power-Analysis Attack - Key Recovery





“Constant-time” means more than just *constant time*

- No branching on secret data: **constant runtime and control flow**

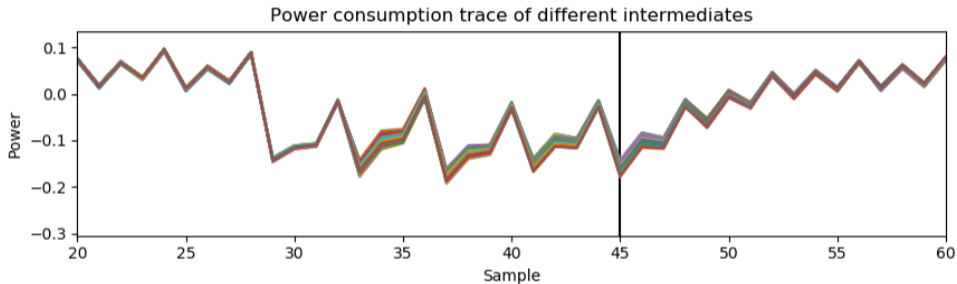
→ always same instruction sequence but different data

- More secure alternatives:
 - Constant-time exponentiation algorithms
 - Constant-time modular reduction
 - ...

And now: “Constant-time” → all problems solved?

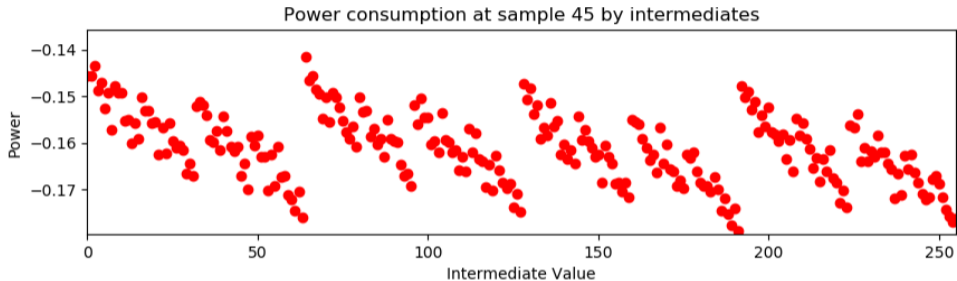
What about data differences?

CMOS Power Consumption: Data Dependency



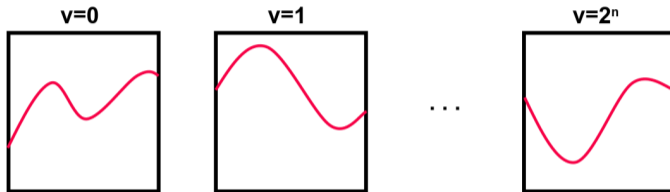
Averaged power traces of a load instruction for values $\{0, 255\}$

CMOS Power Consumption: A Closer Look

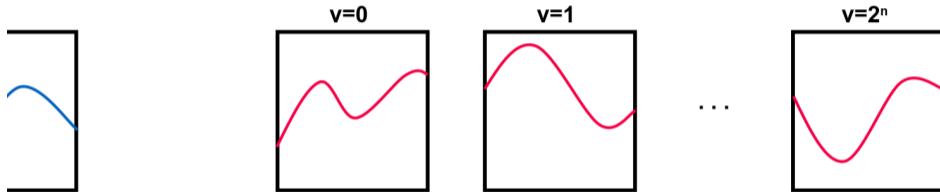


Different intermediate values \rightarrow different power consumption

Record + match values = *Template Attack*



- Profile power consumption for each possible value of intermediate v
- Record traces with all inputs known, group by v
- Profile == “Template”

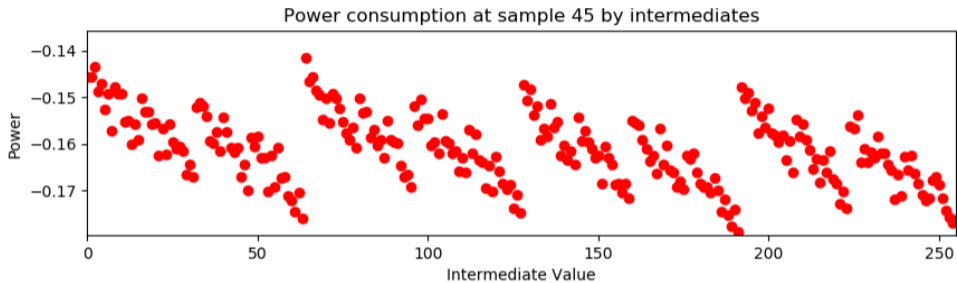


- Compare (match) measured traces to all templates
- Use v which fits best (compute probabilities)



- Pro: Very powerful
 - Key recovery with single trace
 - Sometimes the only option (“we only have a single trace”)
- Contra: many prerequisites and detailed knowledge needed
 - When is secret processed?
 - What is the concrete algorithm?
 - Identical device / setup needed where you can control **all** inputs

Another Look at Intermediate Values

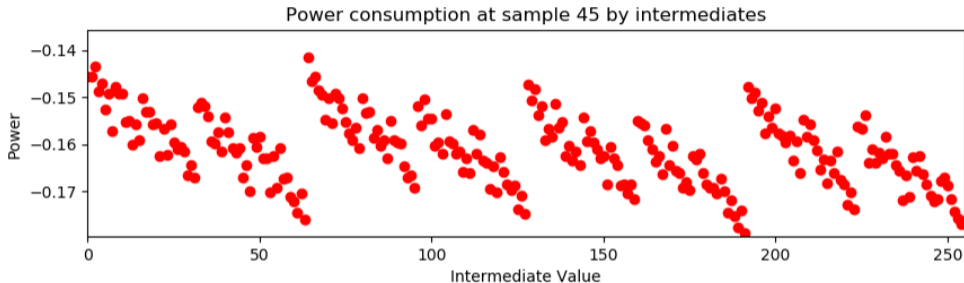


There is some kind of pattern...
We can *model* the power consumption



- “Power consumption depends on switching”
- What’s stored before a value is stored? Assume 0
- Now the new value: each ‘1’-bit draws power → power is proportional to number of bits set
- number of bits set == Hamming weight

And Another Look at Intermediate Values

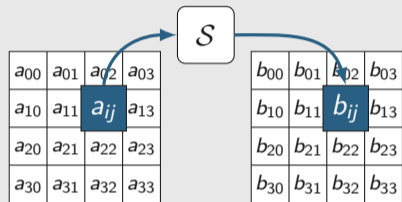


Power

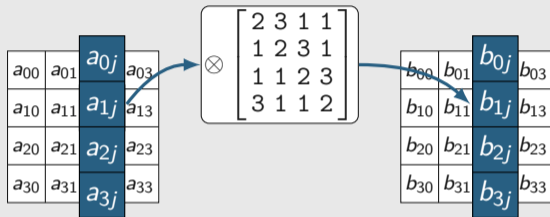
Many devices have similar power behavior → reuse power models
→ an attack without detailed knowledge of device an concrete implementation!

Reminder: AES-128 Block Cipher (10 Rounds)

1. SubBytes (SB)



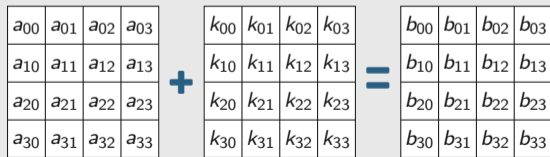
3. MixColumns (MC)

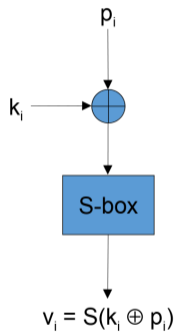


2. ShiftRows (SR)



4. AddRoundKey (AK)

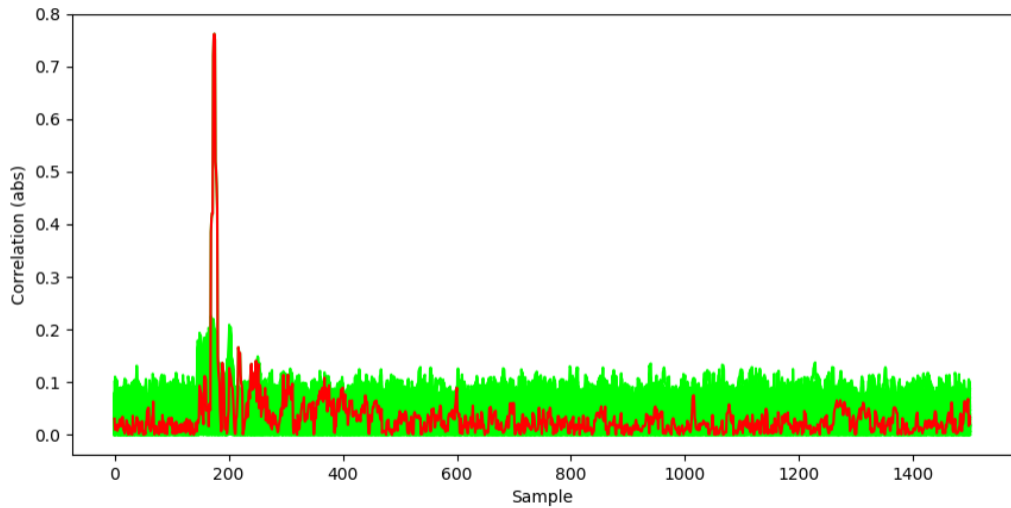




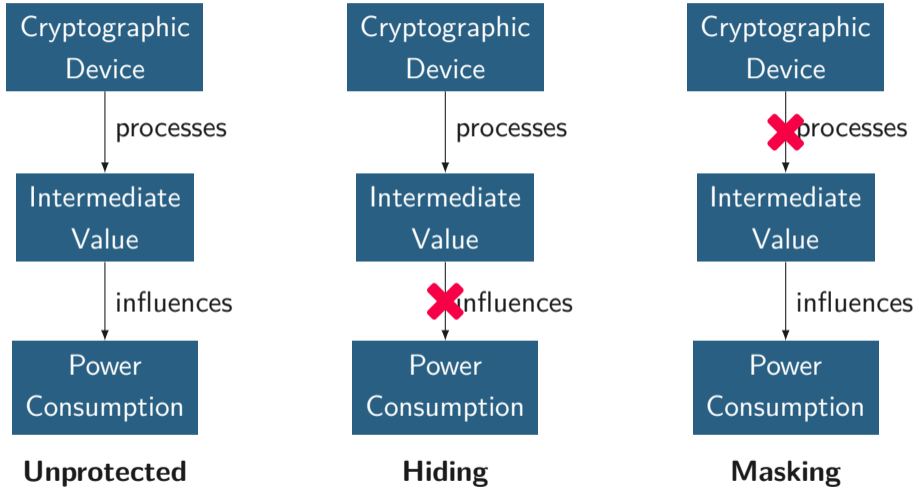
- First round: round key = key
- Other rounds: key schedule
 - key schedule is invertible

Differential Power Analysis (DPA)

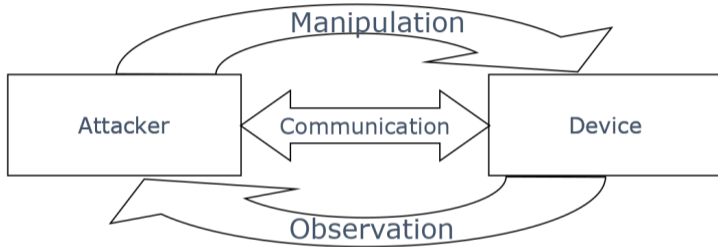
1. **Select intermediate** value that depends on a small number of key bits (subkey)
2. **Measure power** while querying device
3. **Enumerate all possible** subkey **values**
 - 2^8 key hypotheses
 - for each plaintext/ciphertext: predict intermediate for each key hypothesis
4. **Predict power** consumption of intermediate (power model, e.g., Hamming weight)
5. **Compare** prediction with measurement
 - pick key hypothesis that fits best
 - statistical hypothesis tests



Countermeasures against Power Analysis



Fault Attacks

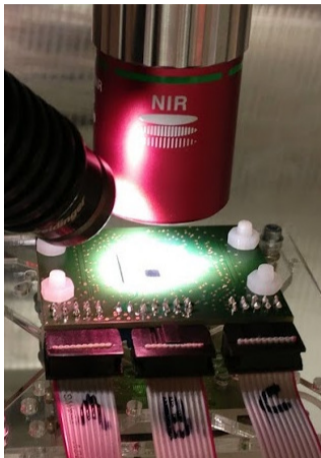


Just listening is boring ...

→ let's manipulate things more actively



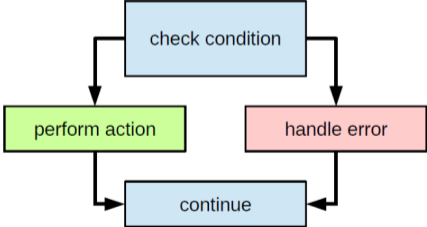
- Goal: manipulate device to compromise security
- Change behavior
 - Deactivate countermeasures / sensors
 - Skip PIN check
- Fault crypto algorithms
 - Compute faulty and correct ciphertexts
 - Use difference to reveal key



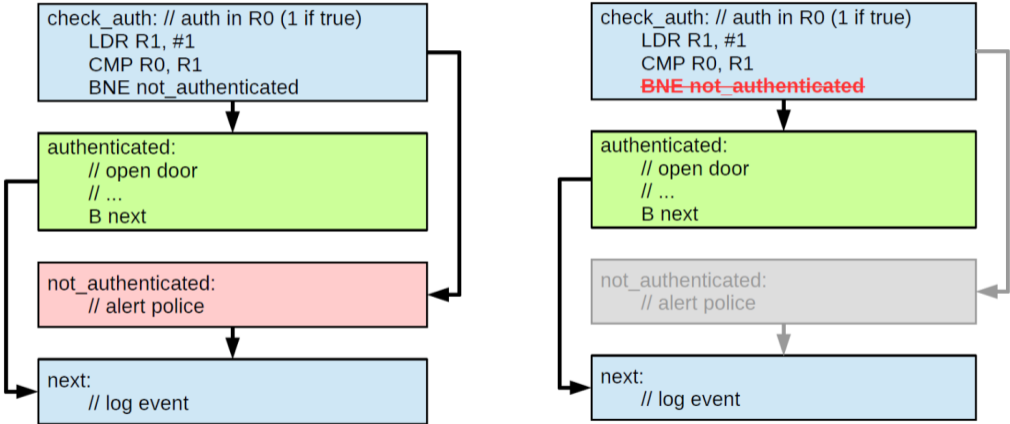
- Spike / glitch: clock, voltage, etc.
- Heat, Radiation, Laser
- Effects:
 - Instructions skipped
 - Data corrupted
 - ...

Example: PIN Check

```
unsigned pin = read_pin();  
bool auth = tpm_check(pin);  
if( auth ) {  
    open_door();  
} else {  
    alert_police();  
}  
log_event();
```



Example: PIN Check - Skipping Attack





PayTV (early 2000s)

- vendors bricked pirated cards via firmware update
- insert endless loop in startup
- solution: glitch to escape loop (“unlooper device”)




Gaming devices

- Xbox360 reset hack
- voltage glitching on reset line
- execute untrusted modified firmware

Fault Attack on RSA

 RSA signatures: $S = M^d \pmod n$, where $n = p \cdot q$

 Efficient implementation trick: Chinese Remainder Theorem (CRT)


 Compute signature result $\pmod p$ and $\pmod q$

$$S_p = S \pmod p = M^{d \bmod p-1} \pmod p$$

$$S_q = S \pmod q = M^{d \bmod q-1} \pmod q$$

 ... and merge the results with CRT:

$$S = S_p \cdot (q^{-1} \bmod p) \cdot q + S_q \cdot (p^{-1} \bmod q) \cdot p \pmod{p \cdot q}$$

 2 exponentiations with half the bit-length and smaller exponents

Fault Attack on RSA Signatures with CRT (“Bellcore attack”)

- Compute signature twice and fault one computation ⚡

$$\begin{aligned} S &= \boxed{S_p \cdot \text{something} \cdot p} \cdot q + \boxed{S_q \cdot \text{some rest} \cdot q} \cdot p \pmod{n} \\ S^\dagger &= \boxed{S_p \cdot (q \text{ faulty} \text{ mod } p)^\dagger} \cdot q + \boxed{S_q \cdot \text{some rest} \cdot q} \cdot p \pmod{n} \\ S - S^\dagger &= \boxed{\text{some garbage}} \cdot q + \boxed{0} \pmod{n} \end{aligned}$$

- Get the secret q using

$$\gcd(S - S^\dagger, n) = \gcd(\boxed{\text{some garbage}} \cdot q, p \cdot q) = q$$

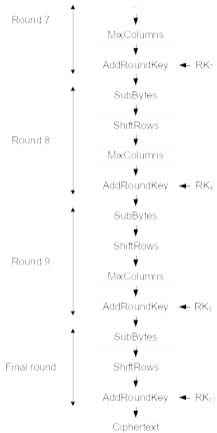
```
bagger> dog Enclave/encl
```

A close-up illustration of Mufasa, the lion king, with a serious and somewhat angry expression. He has a large, reddish-brown mane and yellow eyes. The background is a bright blue sky with white clouds.

MUFARSA

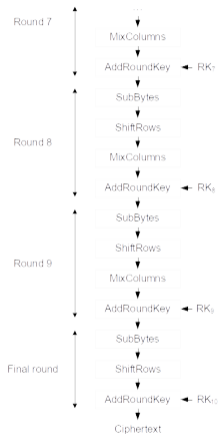
**My Undervolting
Fault Attack on RSA**

Fault Attack on AES



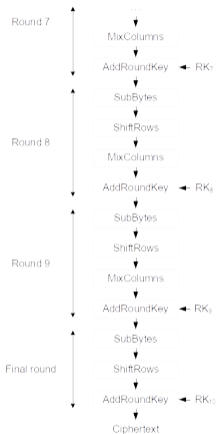
Faulting ciphertext?

- **No.**
- ciphertext difference does not depend on key



Faulting before AddRoundKey10?

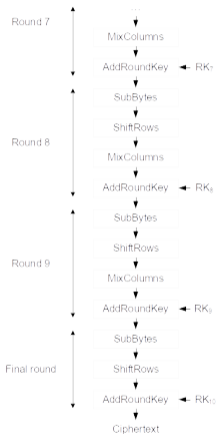
- depends on faults
- not with bit flips (random or known)
- → fault propagates through \oplus :
$$c = v \oplus k$$
$$c' = (v \oplus \Delta v) \oplus k = c \oplus \Delta v$$
- → ciphertext difference still does not depend on key



Faulting before ShiftRows10?

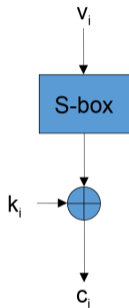
- same as before
- ShiftRows just rearranges bytes

Differential Fault Attacks on AES



Faulting before SubBytes10?

- ...depends on faults
 - Able to flip 1 bit?
- Attacks possible

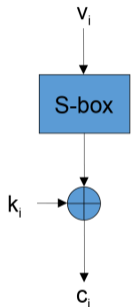


Receive correct and faulty ciphertext

Enumerate all 2^8 values for k_i

- compute back to v (for correct and fault, for all possible k_i)
- compute Δv for k_i
- check if Δv follows ault model (1 bit fault)
- indices can be different because of ShiftRows

Single-Bit Fault before SubBytes - Example



Correct Output = 1A

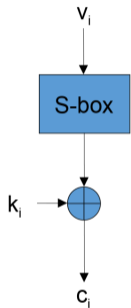
Faulty Output = 99

k: 0 1 2 3 4 5 6 7 8 ...

$C = 1a : S^{-1}(C \text{ xor } k) :$

$C' = 99 : S^{-1}(C' \text{ xor } k) :$

Single-Bit Fault before SubBytes - Example



Correct Output = 1A

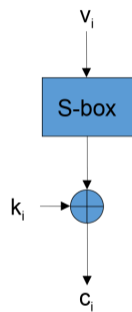
Faulty Output = 99

k: 0 1 2 3 4 5 6 7 8 ...

C = 1a : $S^{-1}(C \text{ xor } k)$: 43 44 34 8e e9 cb c4 de 39 ...

C' = 99 : $S^{-1}(C' \text{ xor } k)$: f9 e2 e8 37 75 1c 6e df ac ...

Single-Bit Fault before SubBytes - Example



Correct Output = 1A

Faulty Output = 99

k: 0 1 2 3 4 5 6 7 8 ...

C = 1a : $S^{-1}(C \text{ xor } k)$: 43 44 34 8e e9 cb c4 de 39 ...

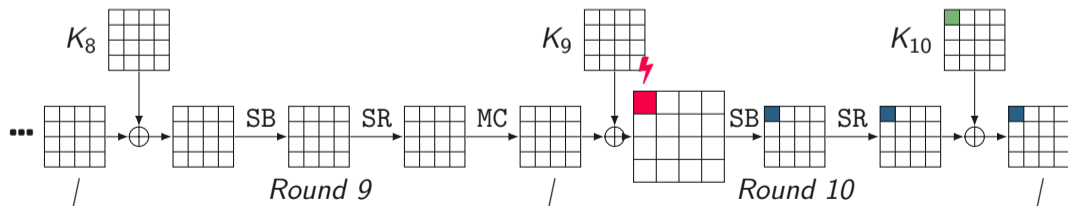
C' = 99 : $S^{-1}(C' \text{ xor } k)$: f9 e2 e8 37 75 1c 6e df ac ...

^^

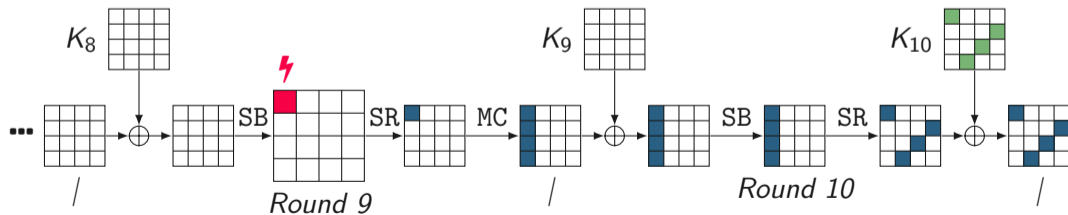
Only few keys have this property \rightarrow filter them

Use further C/C' pairs to get down to 1 key

- Assume the attacker can cause precise **1-bit flips** in Round 9 of AES, before S-box
- For each of 2^8 **key guesses**,
Test if the *partial decryption* produces the expected 1-bit flip.



- Assume the attacker can cause imprecise **1-byte errors**
- For each of 2^{32} **key guesses**,
 Test if the *partial decryption* produces the expected 1-byte error.
 (This can be optimized to require only 2 faulty encryptions to recover the full key)



Instruction	Description
<code>AESENC</code>	Perform one round of an AES encryption flow
<code>AESENCLAST</code>	Perform the last round of an AES encryption flow
<code>AESDEC</code>	Perform one round of an AES decryption flow
<code>AESDECLAST</code>	Perform the last round of an AES decryption flow
<code>AESKEYGENASSIST</code>	Assist in AES round key generation
<code>AESIMC</code>	Assist in AES Inverse Mix Columns
<code>PCLMULQDQ</code>	Carryless multiply (<code>CLMUL</code>)

```
do
{
    i++;
    plaintext = <randomly generated>

    result1 = aes128_enc(plaintext);
    result2 = aes128_enc(plaintext);
} while (vec_equal_128(result1,result2) && i<iterations);
```

```
bagger> sudo ./aes-encrypt 100000 -262
```





**CRYPTO
VAULT
SCREW
HAMMER**

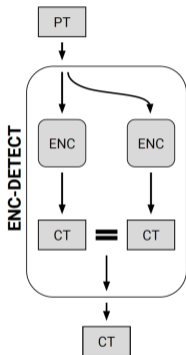
Countermeasures for Fault Attacks



Detect anomalies^a

- Active fine wire meshes across IC → disruption is detected
- Power surge sensors
- Temperature sensors
- Light sensors

^aIBM 4767 Hardware Security Module battery-backed monitoring, meshes, light sensors, temperature sensors, etc. immediate deletion of keying material on tamper detection



Encrypt multiple times, compare result

- comparison at different granularities possible:
- encryption, single round, each operation, ...

But the attacker might be able to

- inject the same fault twice (difficult ...)
- or use more sophisticated methods (statistical attacks)



Attack → defense → next attack → next defense → ...

- different side channels, more sophisticated attacks
- a never-ending cat-and-mouse game

There is no “100% secure”, especially in the physical setting

- any device can be broken by a determined attacker

Our goal:

- Ensure that attack effort is much greater than the value of the secret
- or: Would you do an attack that costs millions to get a free tram ride?

Thanks to Peter Pessl for some of the slides!