

Slides:  
tinyurl.com  
/muujw9ze



# Nothing is Safe

An introduction to hardware (in)security

Nathan Jones



Slides:  
tinyurl.com  
/muujw9ze



## Every device needs hardware security



<https://tinyurl.com/4v5nmrf6>

<https://tinyurl.com/bdew33pb>

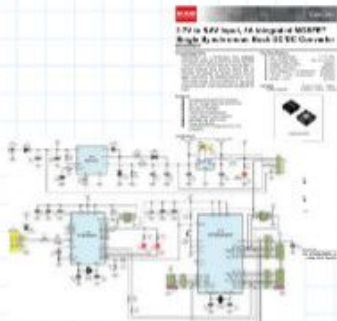


# Contents

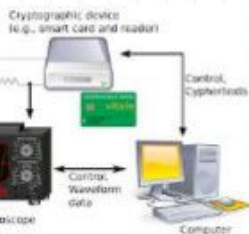
## Level 1: Researching



## Level 2: Snooping

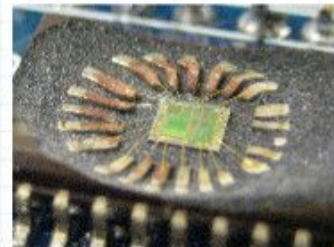
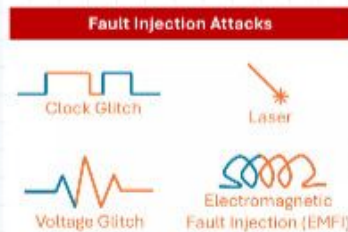


## Level 3: PCB modifications



## Level 4: Side-channel attacks

## Level 5: Fault-injection attacks



## Level 6: IC decapping

Slides:  
tinyurl.com  
/muujw9ze



**Nathan Jones**



**HARVEY  
MUDD  
COLLEGE**

**NC STATE  
UNIVERSITY**

**DigiKey**

Selected publications:

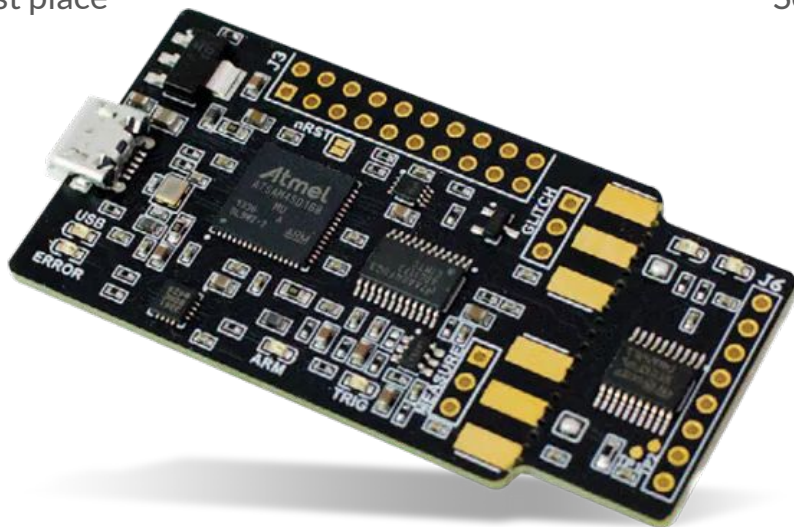
- ["Inside the Voja4"](#) (Hackaday Supercon)
- ["OOP in C"](#) (Embedded Online Conference)
- ["You Don't Need an RTOS"](#) (EmbeddedRelated.com)
- ["Message Passing with Pointers"](#) (Maker.io)

Slides:  
tinyurl.com  
/muujw9ze

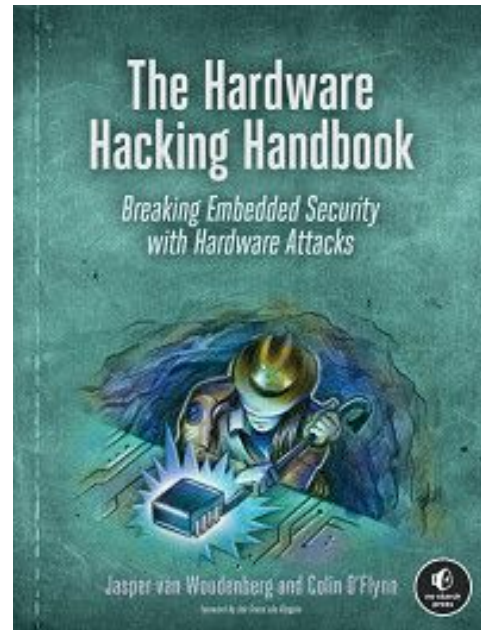


## Prizes!

First place



Second and Third Places



Slides:  
tinyurl.com  
/muujw9ze



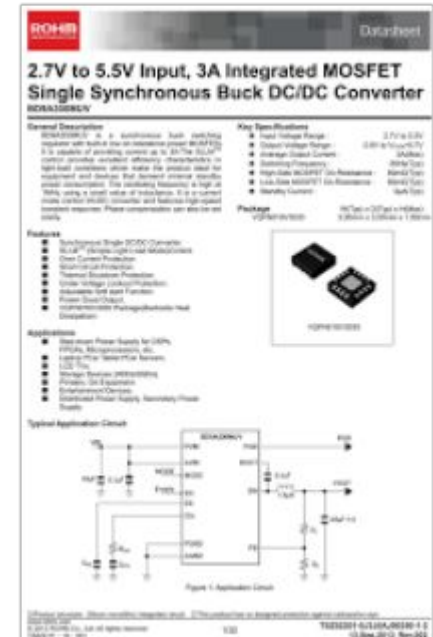
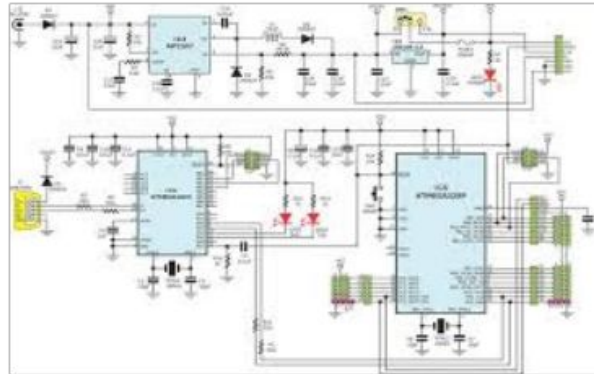
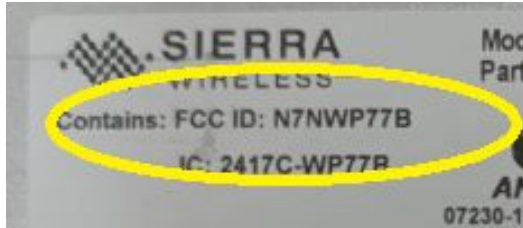
---

# Level 1: Researching



# What information can be found on the internet?

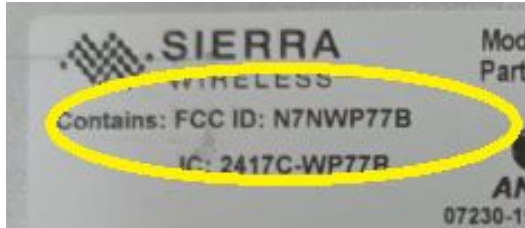
- Product manuals
- Schematics and firmware
- (Datasheets for internal components)
- Repair tutorials
- Patent filings
- FCC filings



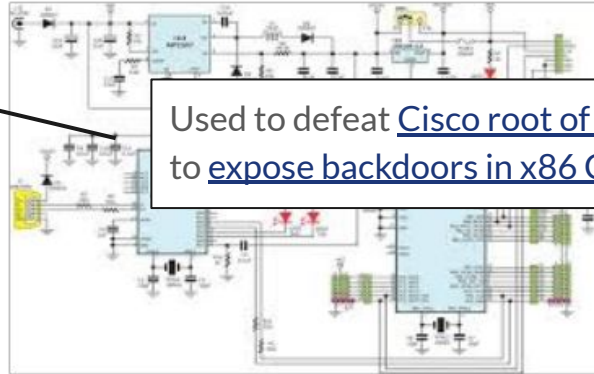


# What information can be found on the internet?

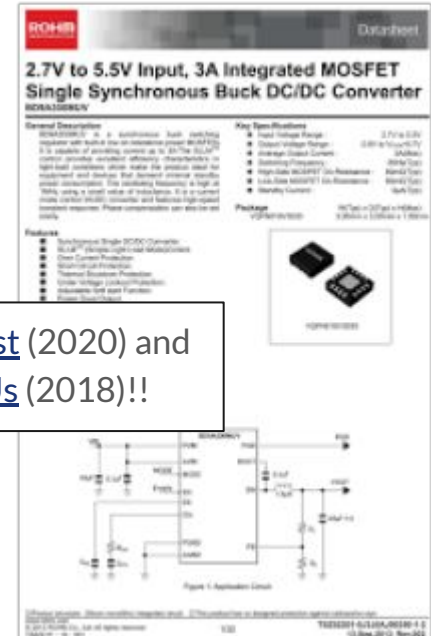
- Product manuals
- Schematics and firmware
- (Datasheets for internal components)
- Repair tutorials
- Patent filings
- FCC filings



[https://cdn.kthlaw.com/media/public/content/images/fcc\\_id.png](https://cdn.kthlaw.com/media/public/content/images/fcc_id.png)



Used to defeat [Cisco root of trust](#) (2020) and to [expose backdoors in x86 CPUs](#) (2018)!!





- 

[https://www.autodesk.com/products/fusion-360/blog/wp-content/uploads/2021/08/cyber-2377718\\_1280-1024x686.jpg](https://www.autodesk.com/products/fusion-360/blog/wp-content/uploads/2021/08/cyber-2377718_1280-1024x686.jpg)

Slides:  
tinyurl.com  
/muujw9ze



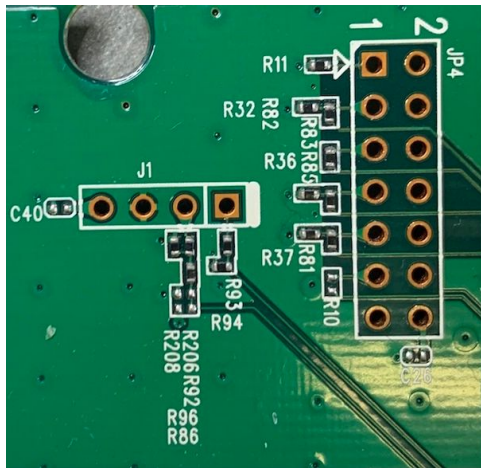
---

# Level 2: Snooping

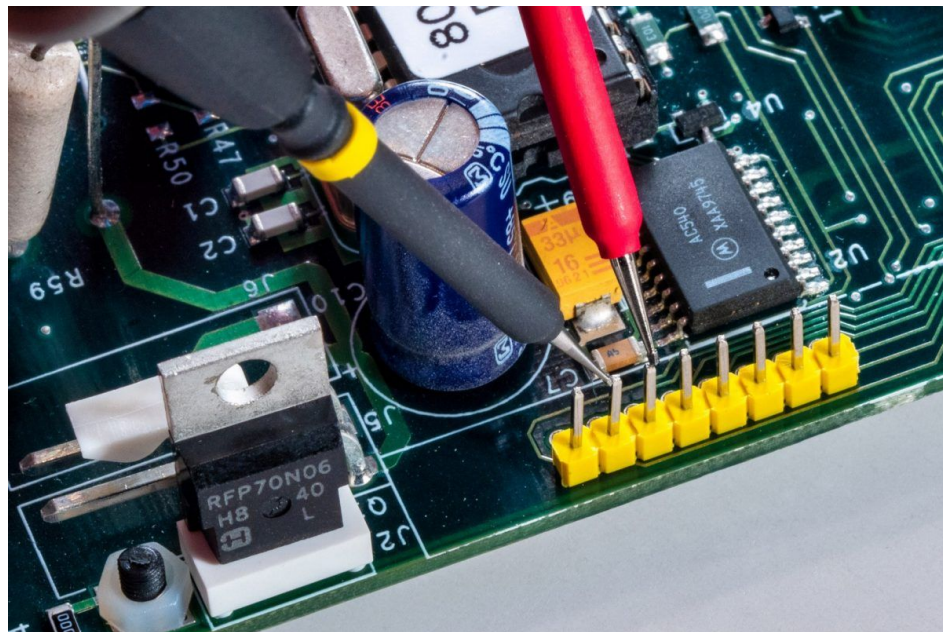


# What signals are being transmitted during operation?

- JTAG, UART, SPI, etc



<https://riverloopsecurity.com/img/blog/hw-101-jtag-header.jpg>

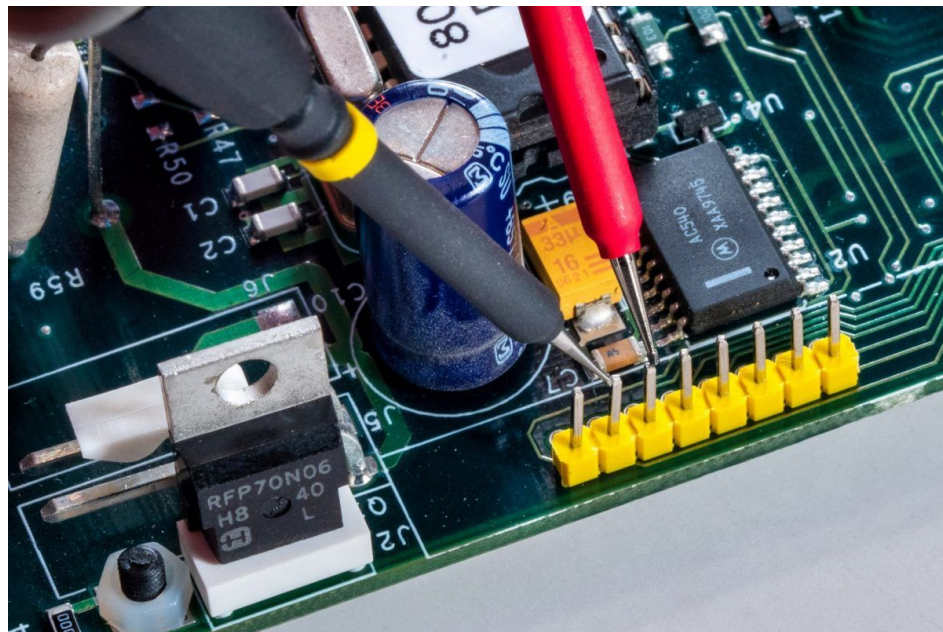


<https://www.tag-connect.com/wp-content/uploads/2018/03/no-slip-probes-2-1200x800.jpg>



# What signals are being transmitted during operation?

- Is the JTAG port unlocked?
- Is a decrypted firmware image being transferred from external Flash to the main CPU upon start-up?
- What signals are used to interface with any peripherals/sensors/actuators?
- How *long* does it take the UART shell to reply "Access denied" after a failed login attempt?

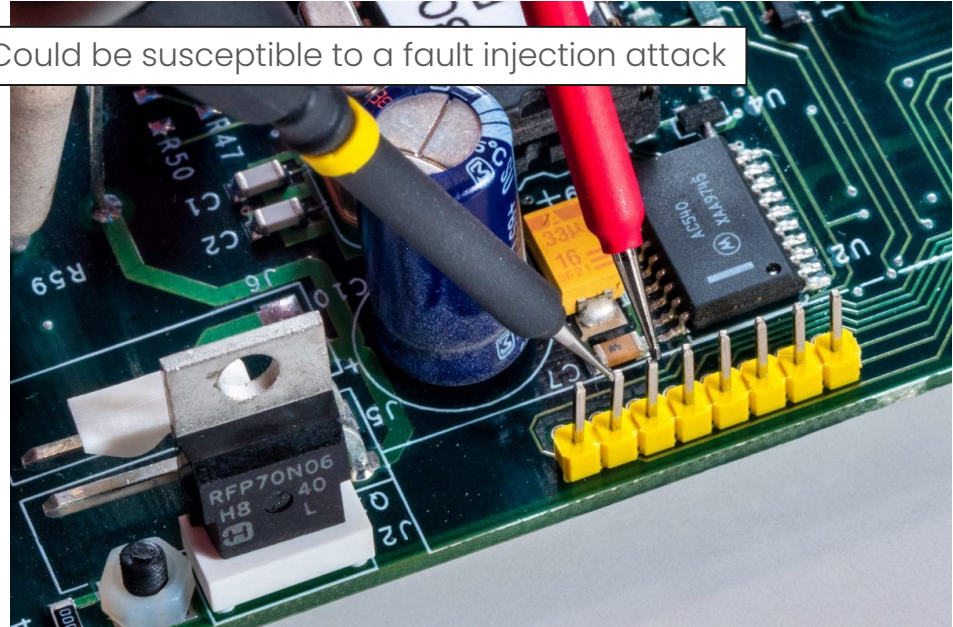




# What signals are being transmitted during operation?

- Is the JTAG port unlocked? ←
- Is a decrypted firmware image being transferred from external Flash to the main CPU upon start-up?
- What signals are used to interface with any peripherals/sensors/actuators?
- How *long* does it take the UART shell to reply "Access denied" after a failed login attempt?

Could be susceptible to a fault injection attack



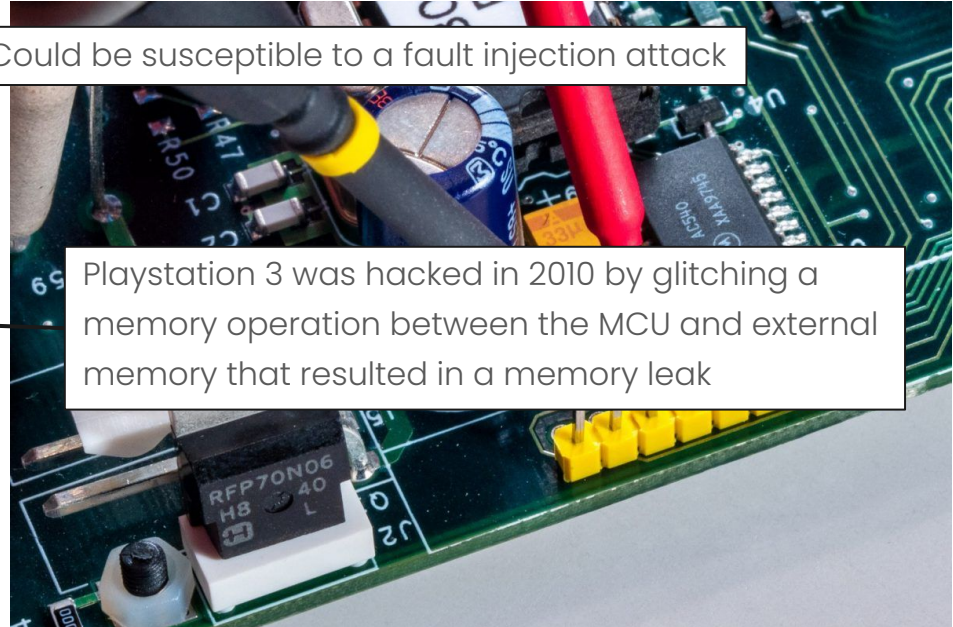


# What signals are being transmitted during operation?

- Is the JTAG port unlocked? ←
- Is a decrypted firmware image being transferred from external Flash to the main CPU upon start-up?
- What signals are used to interface with any peripherals/sensors/actuators? ←
- How *long* does it take the UART shell to reply "Access denied" after a failed login attempt?

Could be susceptible to a fault injection attack

Playstation 3 was hacked in 2010 by glitching a memory operation between the MCU and external memory that resulted in a memory leak





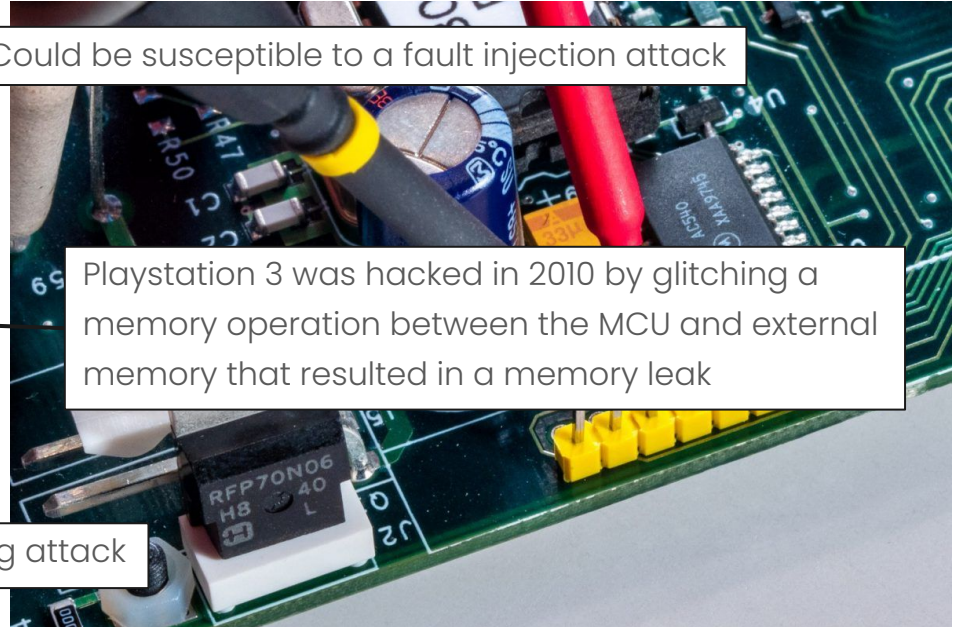
# What signals are being transmitted during operation?

- Is the JTAG port unlocked?
- Is a decrypted firmware image being transferred from external Flash to the main CPU upon start-up?
- What signals are used to interface with any peripherals/sensors/actuators?
- How *long* does it take the UART shell to reply "Access denied" after a failed login attempt?

Could be susceptible to a fault injection attack

Playstation 3 was hacked in 2010 by glitching a memory operation between the MCU and external memory that resulted in a memory leak

Could permit a timing attack



Slides:  
tinyurl.com  
/muujw9ze



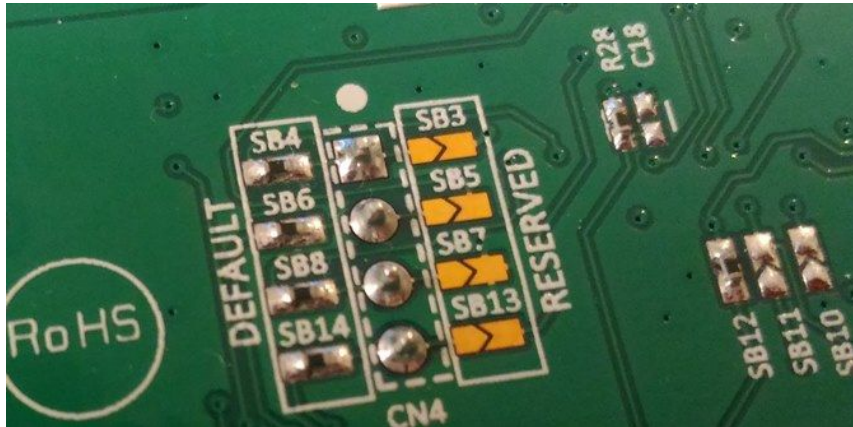
---

# Level 3: PCB modifications

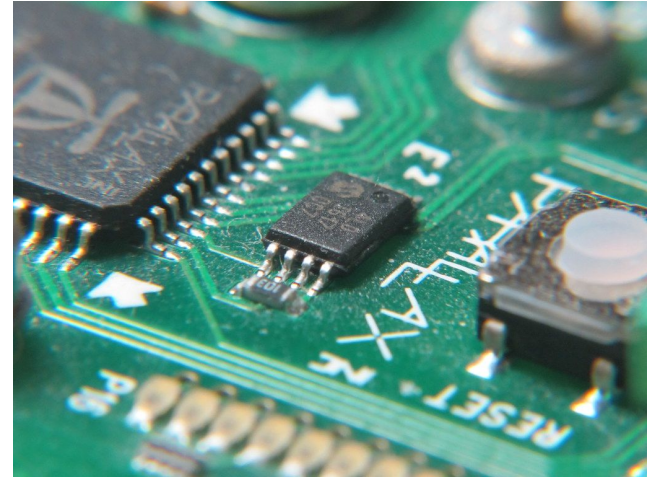


## Physically alter the circuit

- Configuration resistors/jumpers
- Read from/write to external memory



<https://i.sstatic.net/iis7l.jpg>



[https://www.totalphase.com/media/blog/2020/06/4653066327\\_e29062b4b7\\_b.jpg](https://www.totalphase.com/media/blog/2020/06/4653066327_e29062b4b7_b.jpg)

Slides:  
tinyurl.com  
/muujw9ze



# Discuss!

What other attacks could be enabled or conducted by a person who simply conducts some research or has access to the internal PCB?

If it helps, think of how you might begin attacking device that you've built or that you own!

Slides:  
tinyurl.com  
/muujw9ze



# Some hardware attacks aren't all that difficult to conduct.

---

Slides:  
tinyurl.com  
/muujw9ze



**Kahoot!**

Slides:  
tinyurl.com  
/muujw9ze



---

# Level 4: Side-channel attacks



## What's a side-channel?

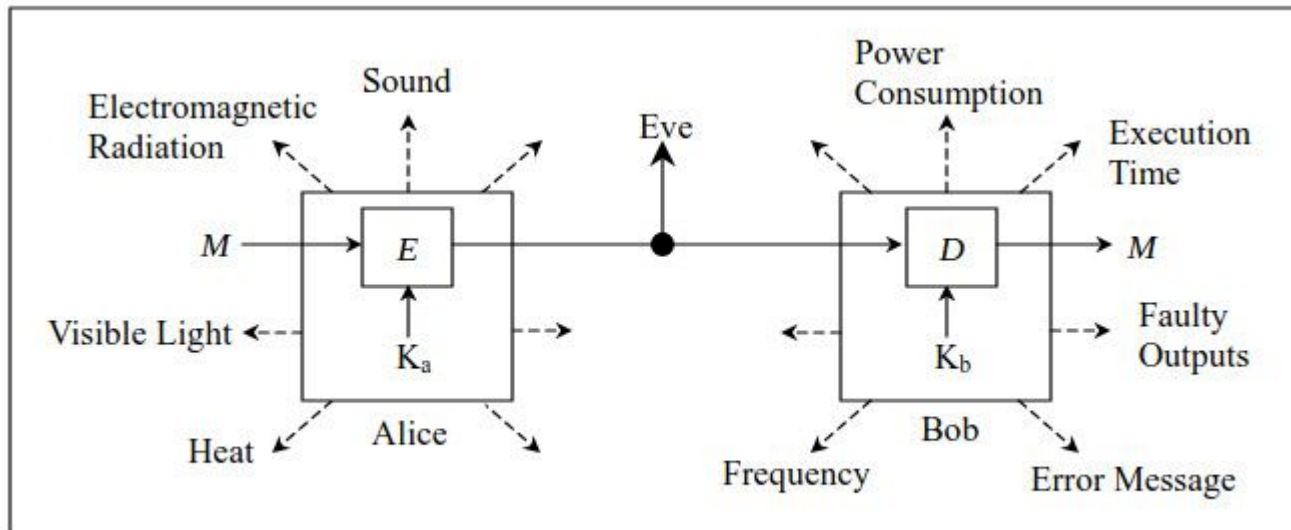


Figure 2: The cryptographic model including side-channel

Slides:  
tinyurl.com  
/muujw9ze

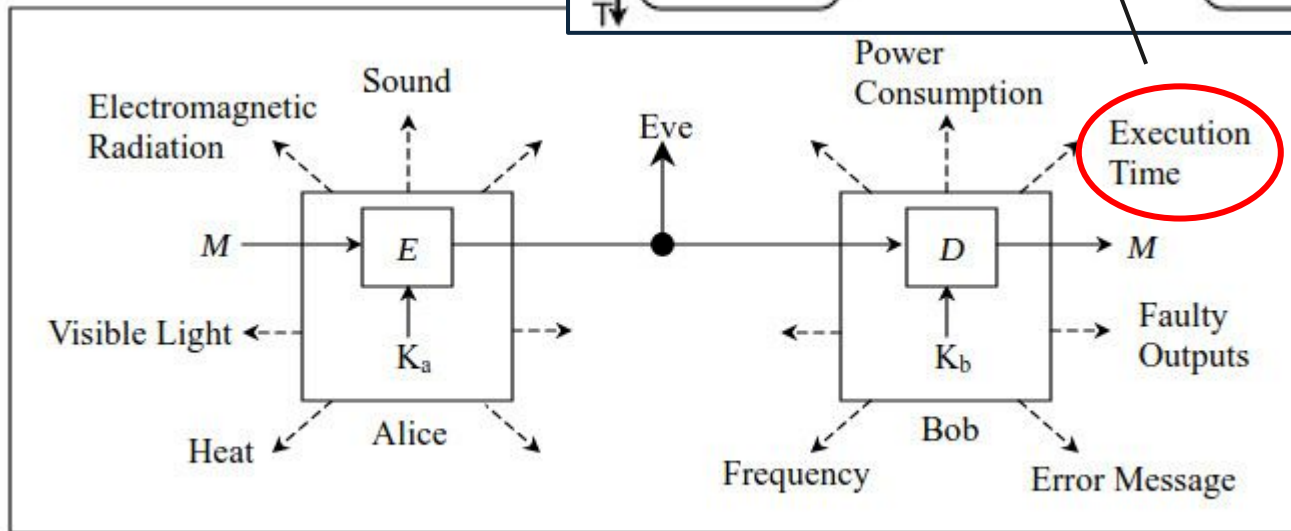
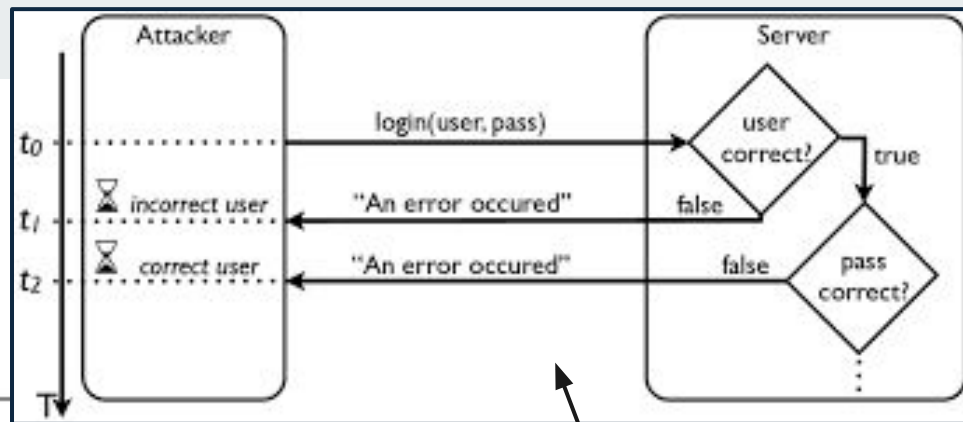


# Every device leaks information about its operations and data.

---



## Timing attacks



<https://tinyurl.com/4x6a3bcy>

Figure 2: The cryptographic model including side-channel

<https://eprint.iacr.org/2005/388.pdf>

Slides:  
tinyurl.com  
/muujw9ze



# Timing attacks on strcmp

## Host computer

```
>> Enter password:
```

## Password-protected device

```
char * correctPw = "123456";

if(strcmp(enteredPw, correctPw) == 0)
{
    printf("Access granted! :) \n");
}
else
{
    printf("Access denied :( \n");
}
```



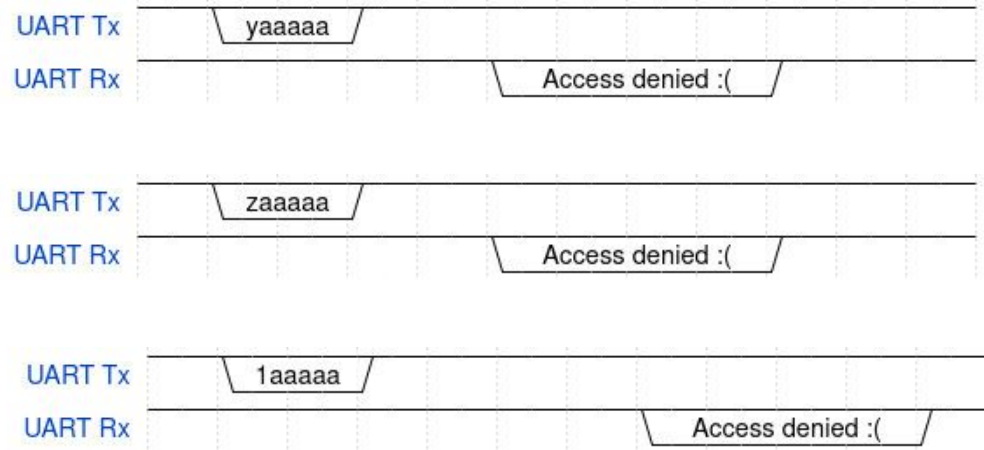
# Timing attack on strcmp

## Host computer

```
>> Enter password: yaaaaa  
Access denied :(
```

```
>> Enter password: zaaaaa  
Access denied :(
```

```
>> Enter password: laaaaa  
Access denied :(
```



Aha!!

Slides:  
tinyurl.com  
/muujw9ze



# 1,088,391,168

Number of passwords to brute-force (assuming passwords must be 6 lower-case or numeric characters)

---

Slides:  
tinyurl.com  
/muujw9ze



# 108

Number of passwords to brute-force using timing attack

---

Slides:  
tinyurl.com  
/muujw9ze



## Timing attack on a pin-protected HDD



<https://c1.neweggimages.com/productimage/nb640/17-392-033-12.jpg>

Slides:  
tinyurl.com  
/muujw9ze



# Discuss!

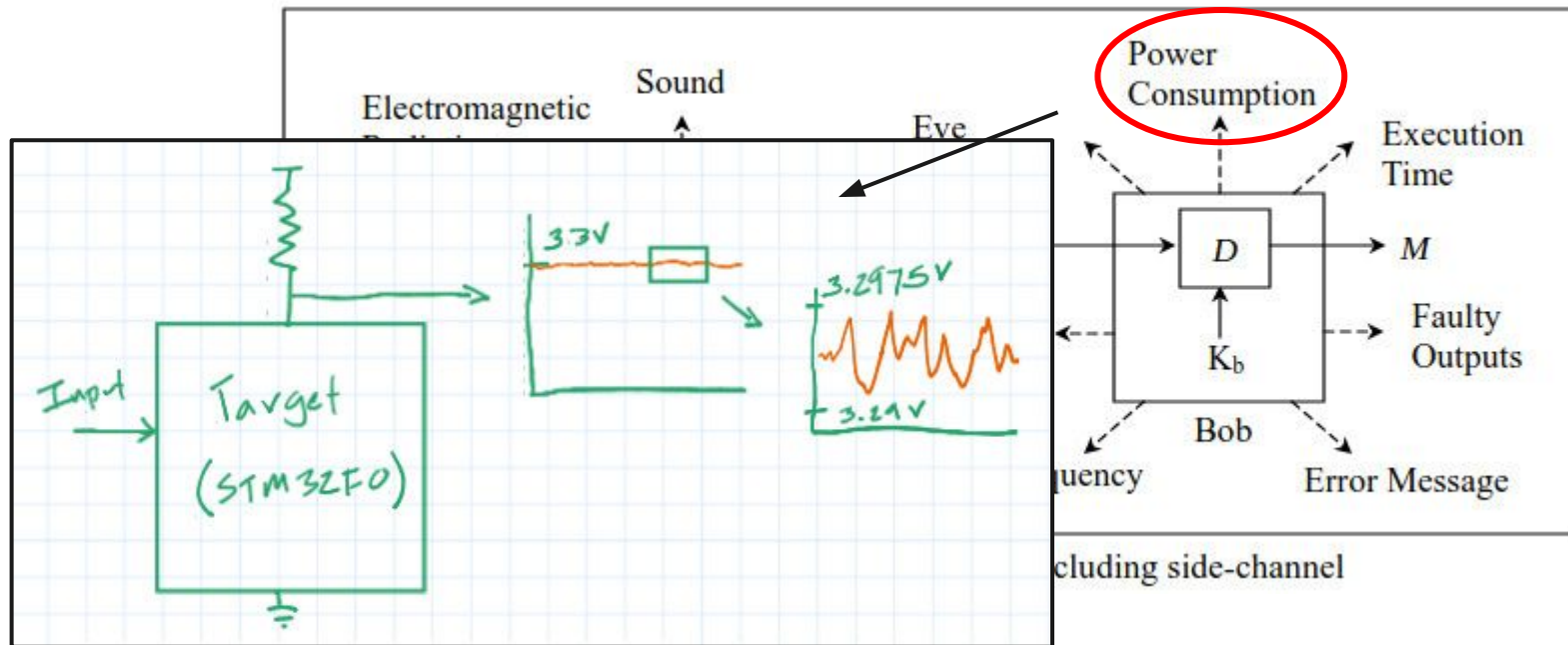
How would you fix the code?

```
char * correctPw = "123456";

if(strcmp(enteredPw, correctPw) == 0)
{
    printf("Access granted! :) \n");
}
else
{
    printf("Access denied :( \n");
}
```

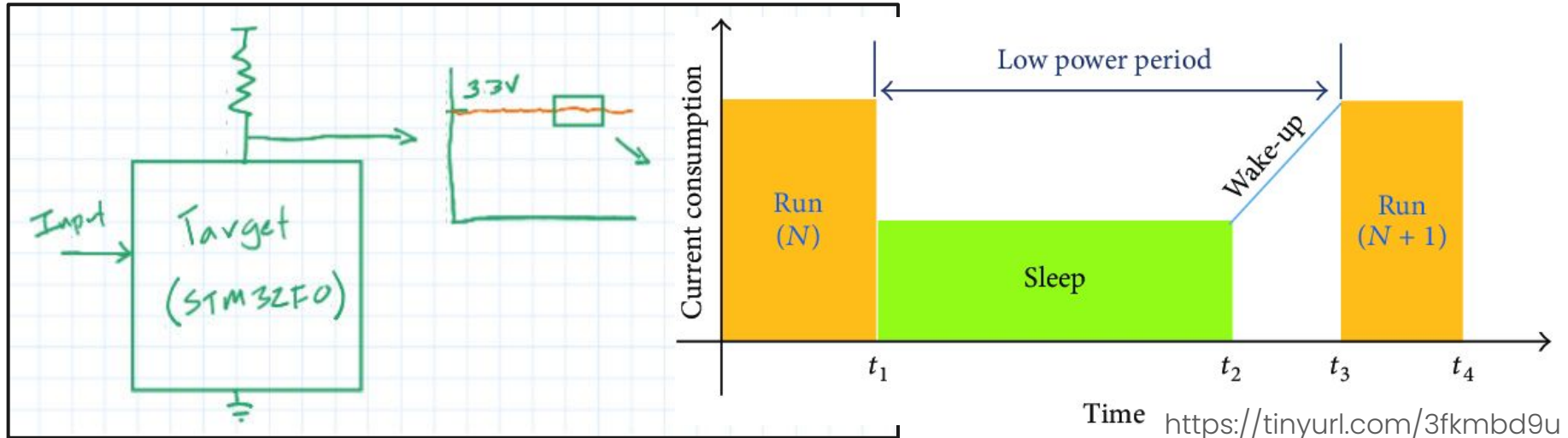


## Power analysis attacks



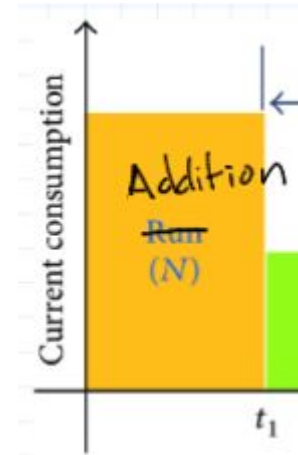
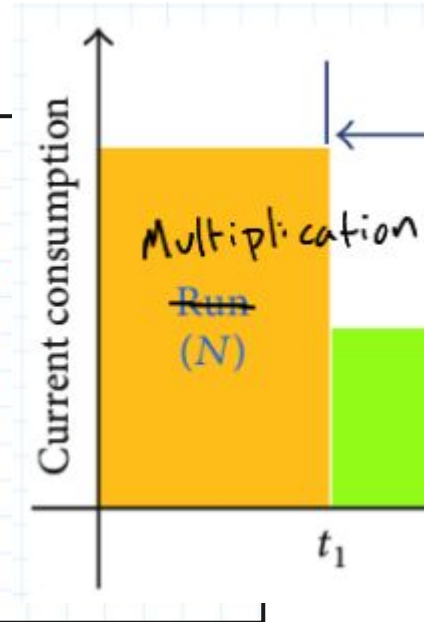
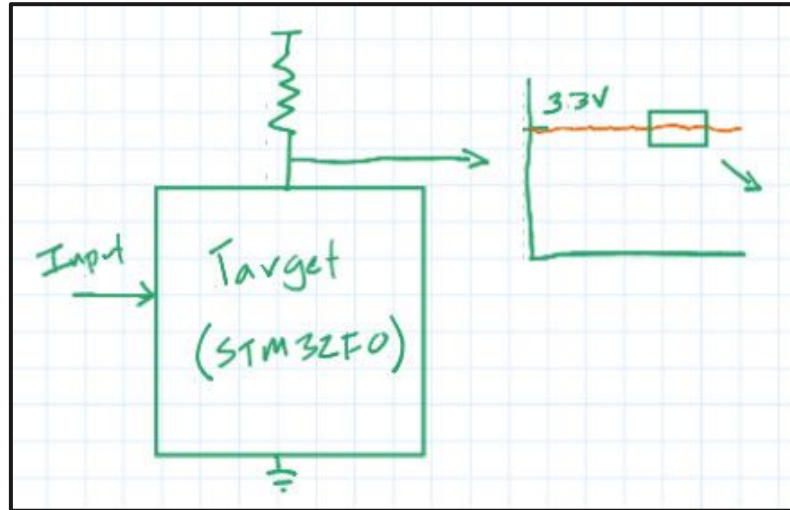


## Power analysis can show: active vs sleep modes



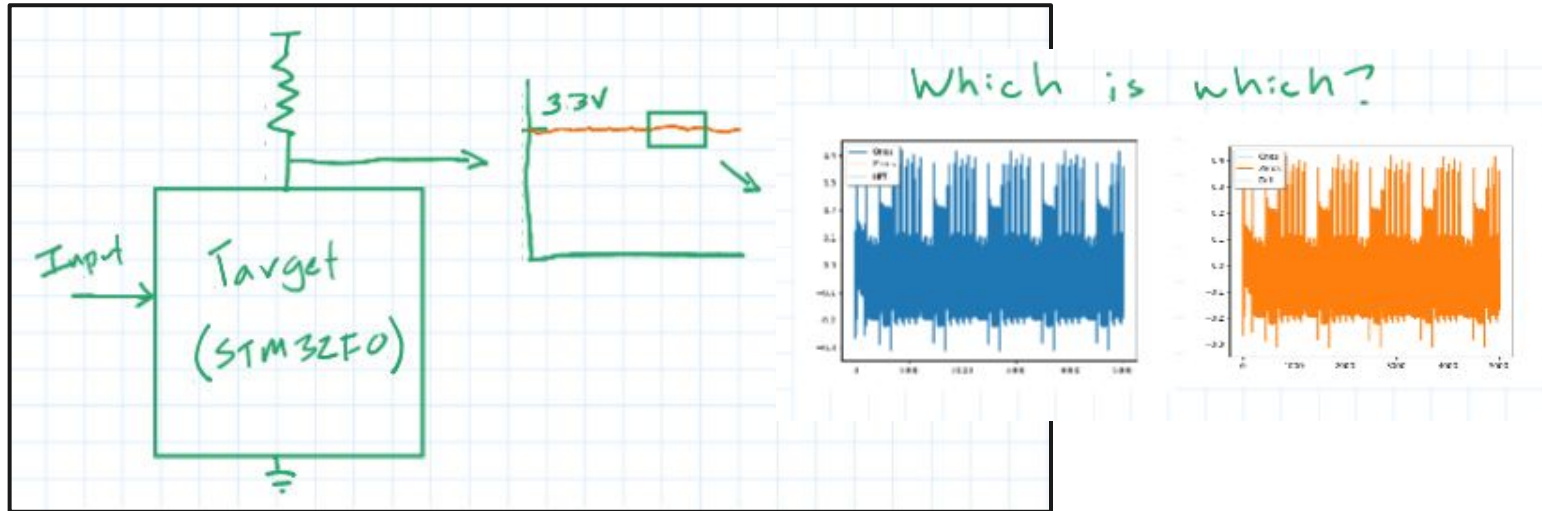


## Power analysis can show: one operation vs another



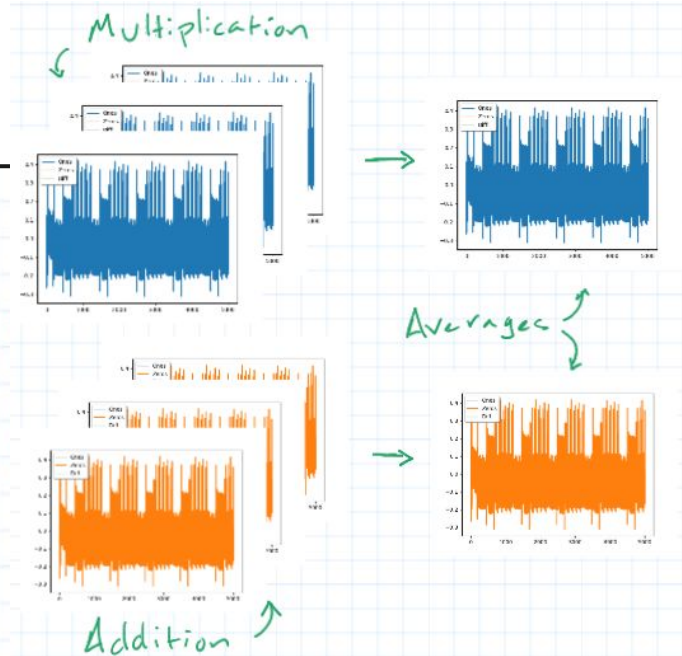
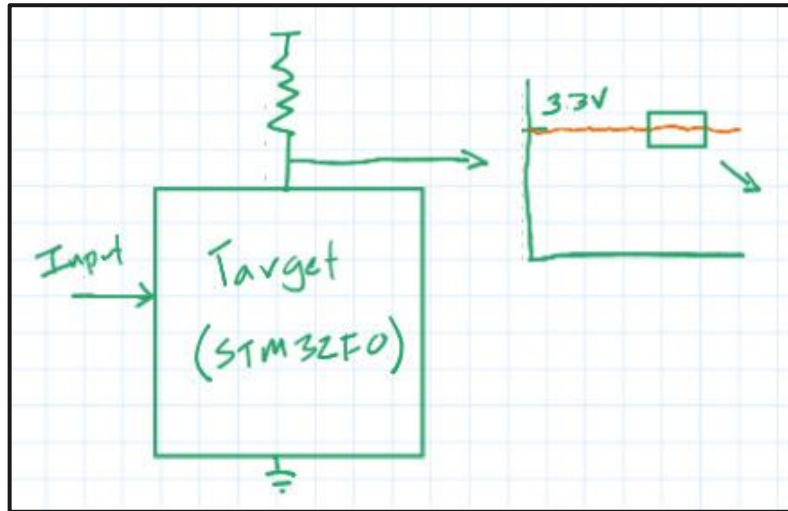


## Identifying a single bit in a single trace is hard...





...so we'll use statistics (difference of means [DoM])!

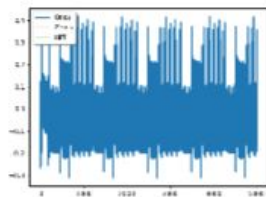


Slides:  
tinyurl.com  
/muujw9ze

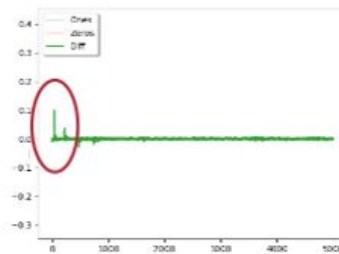
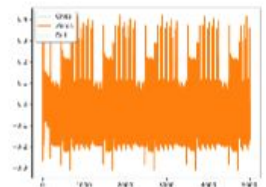


## DoM shows tiny differences between traces

Avg multiplication

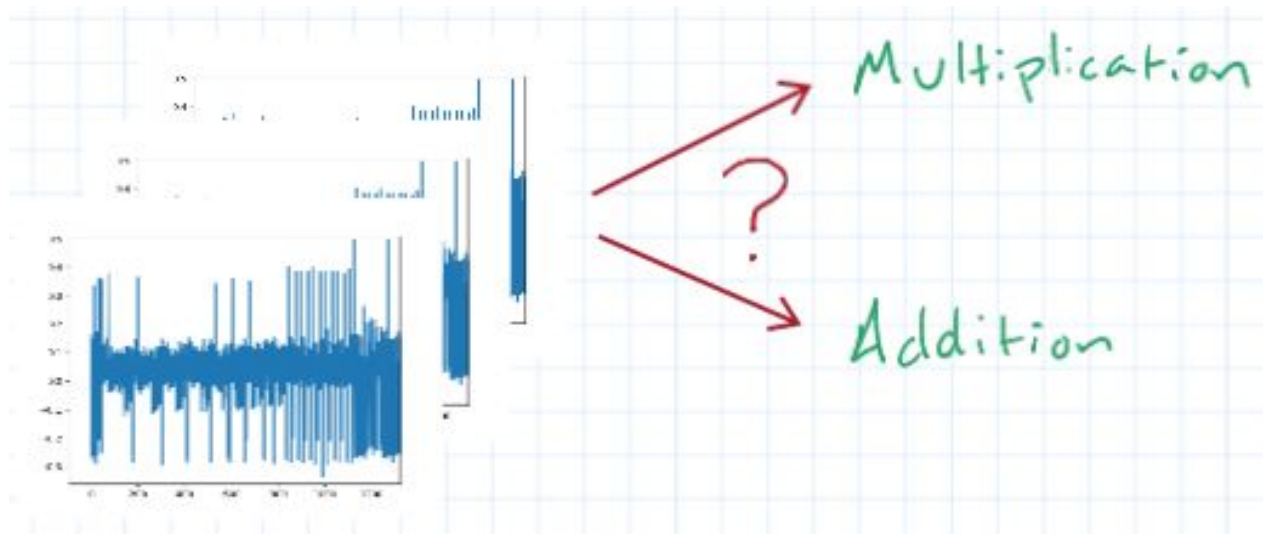


Avg addition



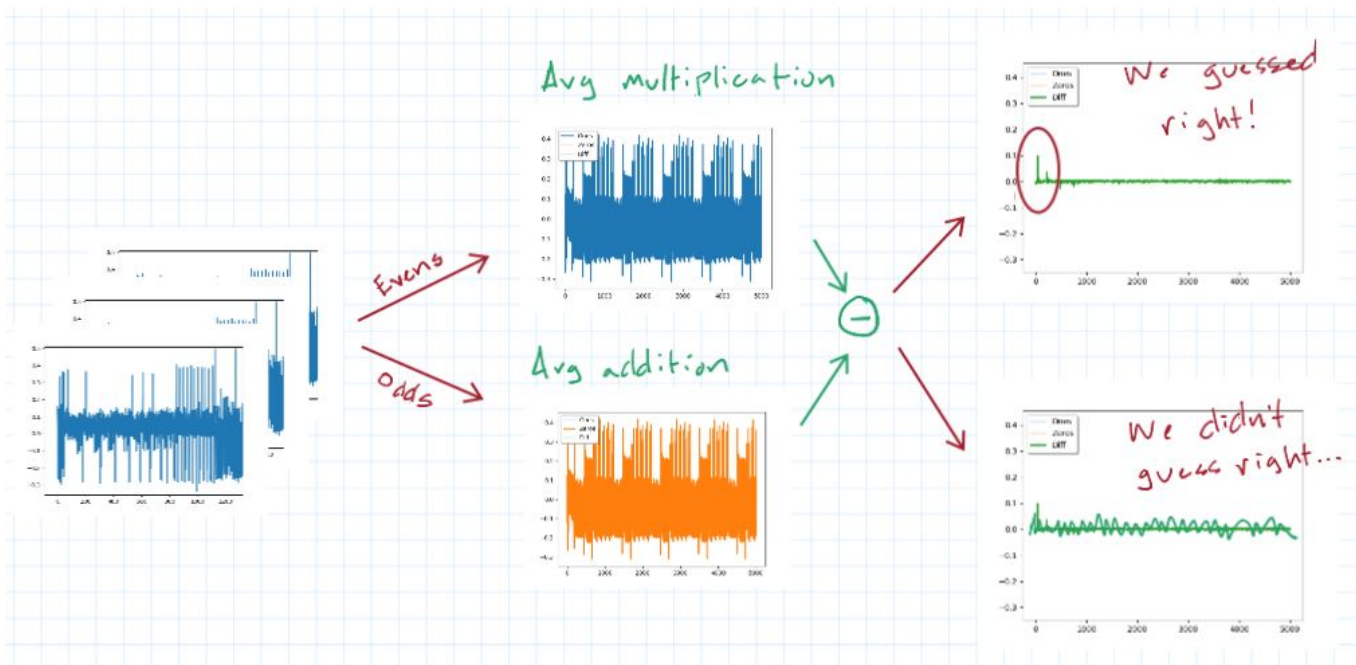


If we didn't *know* which trace was which, we'd need to guess...





...and then check.



Slides:  
tinyurl.com  
/muujw9ze



**Power analysis can show:**  
*whether a single bit in the  
microcontroller is a 1 or a 0.*

---

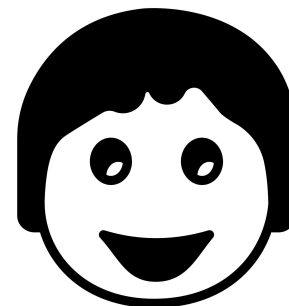
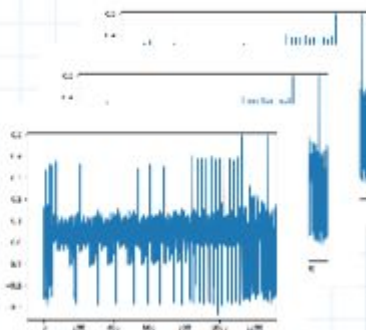
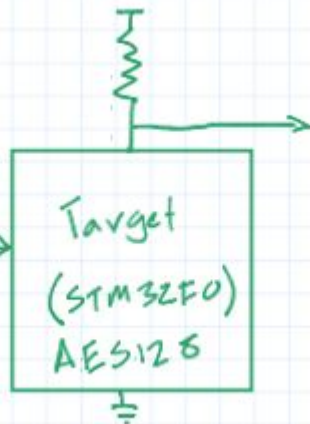


## Power analysis attack on AES128

Ah! The private key  
must be 0x2B7F15...

[0x34, 0xA0, ...]  
[0x7D, 0x06, ...]  
[0x0C, 0xF1, ...]  
[0xB5, 0x82, ...]

Input



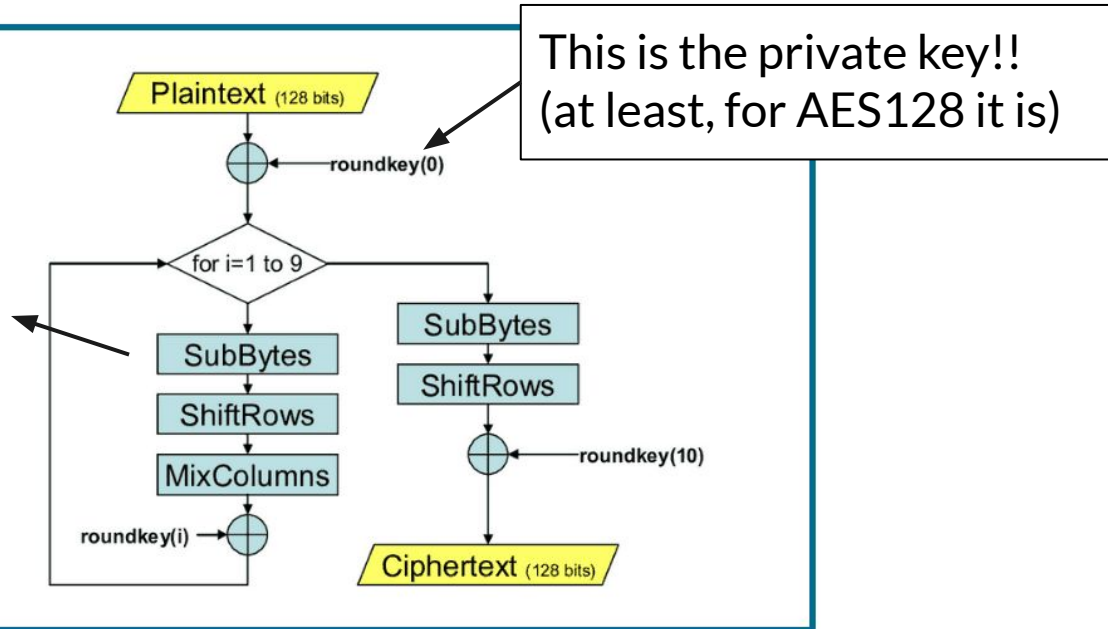


## A brief overview of AES

AES S-box

|    | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

[https://en.wikipedia.org/wiki/Rijndael\\_S-box](https://en.wikipedia.org/wiki/Rijndael_S-box)



<https://privacycanada.net/app/uploads/2019/01/aes-algorithm-diagram.png>

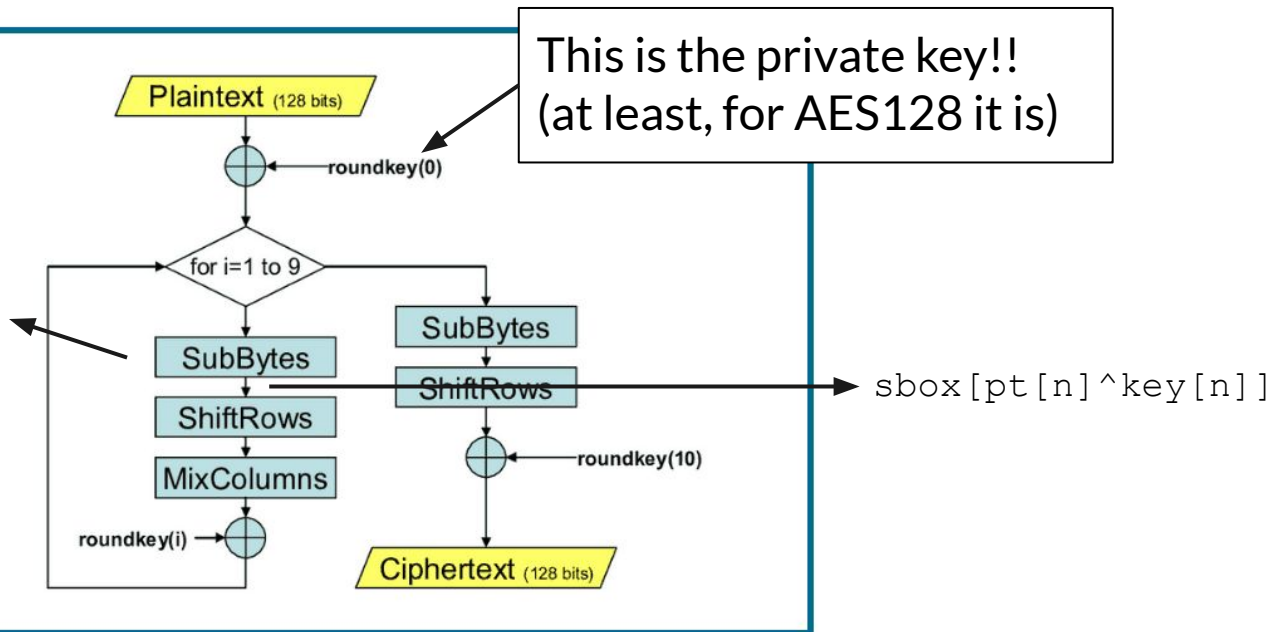


## A brief overview of AES

AES S-box

|    | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

[https://en.wikipedia.org/wiki/Rijndael\\_S-box](https://en.wikipedia.org/wiki/Rijndael_S-box)



<https://privacycanada.net/app/uploads/2019/01/aes-algorithm-diagram.png>

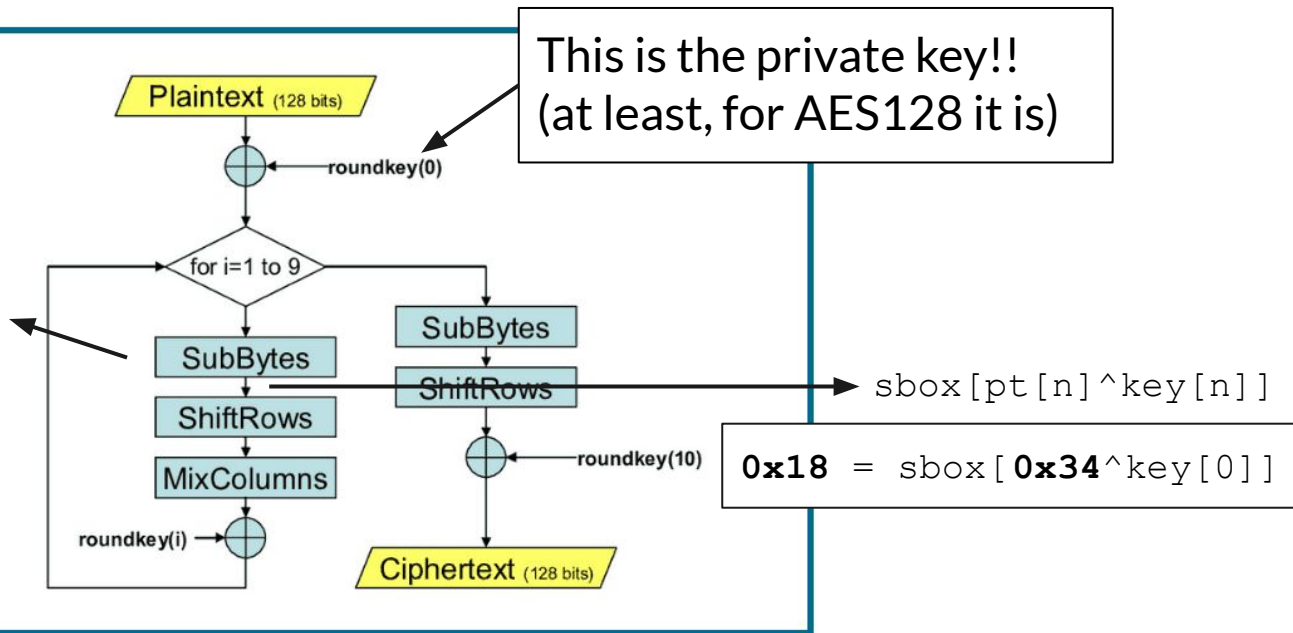


## Recovering an AES key

AES S-box

|    | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

[https://en.wikipedia.org/wiki/Rijndael\\_S-box](https://en.wikipedia.org/wiki/Rijndael_S-box)



<https://privacycanada.net/app/uploads/2019/01/aes-algorithm-diagram.png>

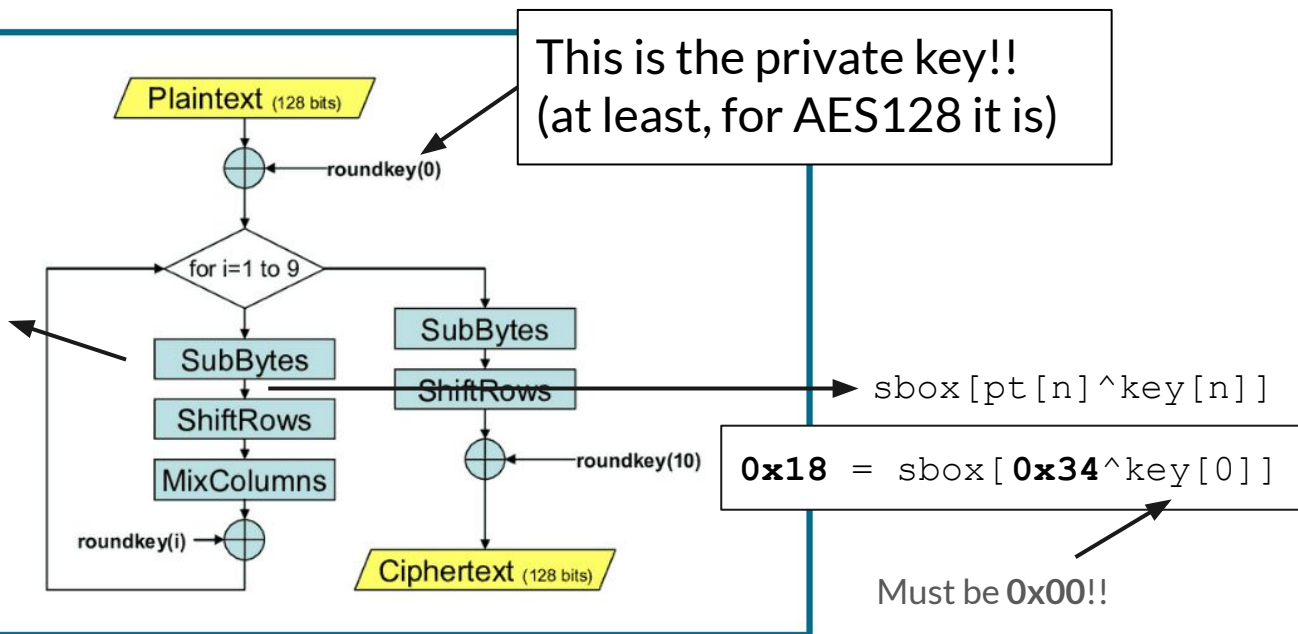


## Recovering an AES key

AES S-box

|    | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

[https://en.wikipedia.org/wiki/Rijndael\\_S-box](https://en.wikipedia.org/wiki/Rijndael_S-box)



<https://privacycanada.net/app/uploads/2019/01/aes-algorithm-diagram.png>

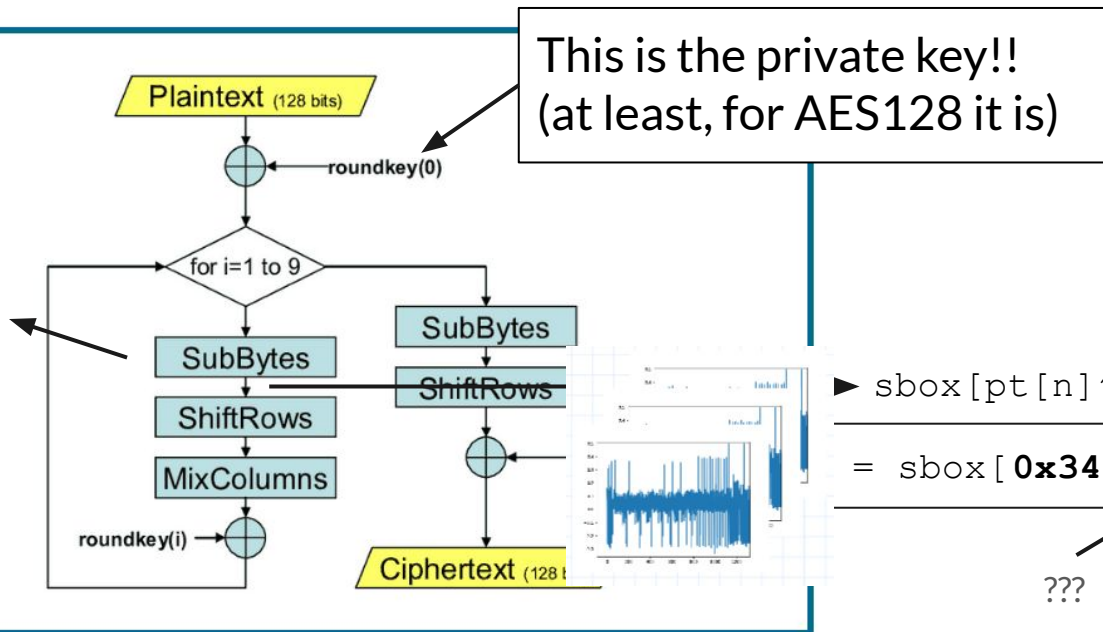


## Recovering an AES key

AES S-box

|    | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

[https://en.wikipedia.org/wiki/Rijndael\\_S-box](https://en.wikipedia.org/wiki/Rijndael_S-box)



$$\begin{aligned} & \text{sbox}[pt[n]^key[n]] \\ &= \text{sbox}[0x34^key[0]] \end{aligned}$$

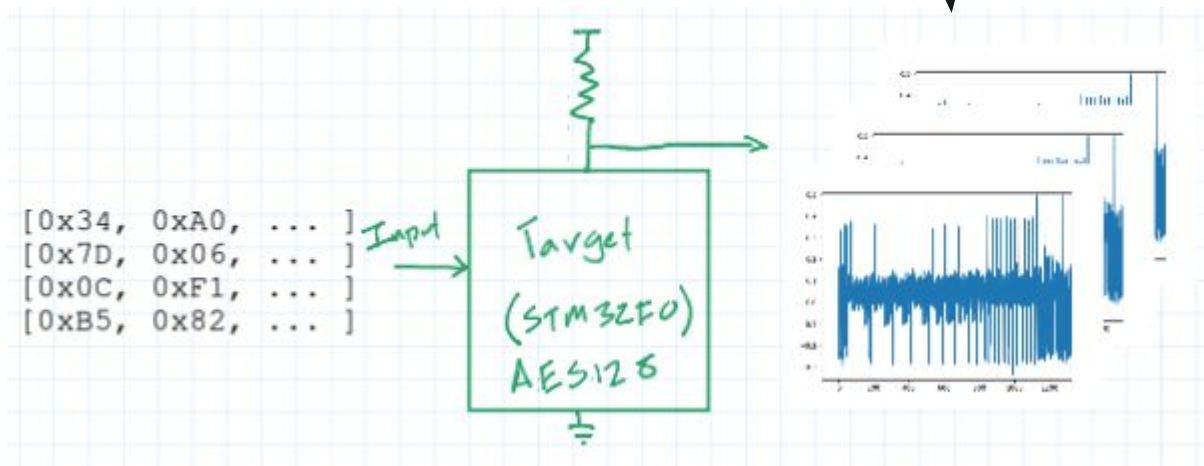
???

<https://privacycanada.net/app/uploads/2019/01/aes-algorithm-diagram.png>



# Differential Power Analysis (DPA) attack on AES128

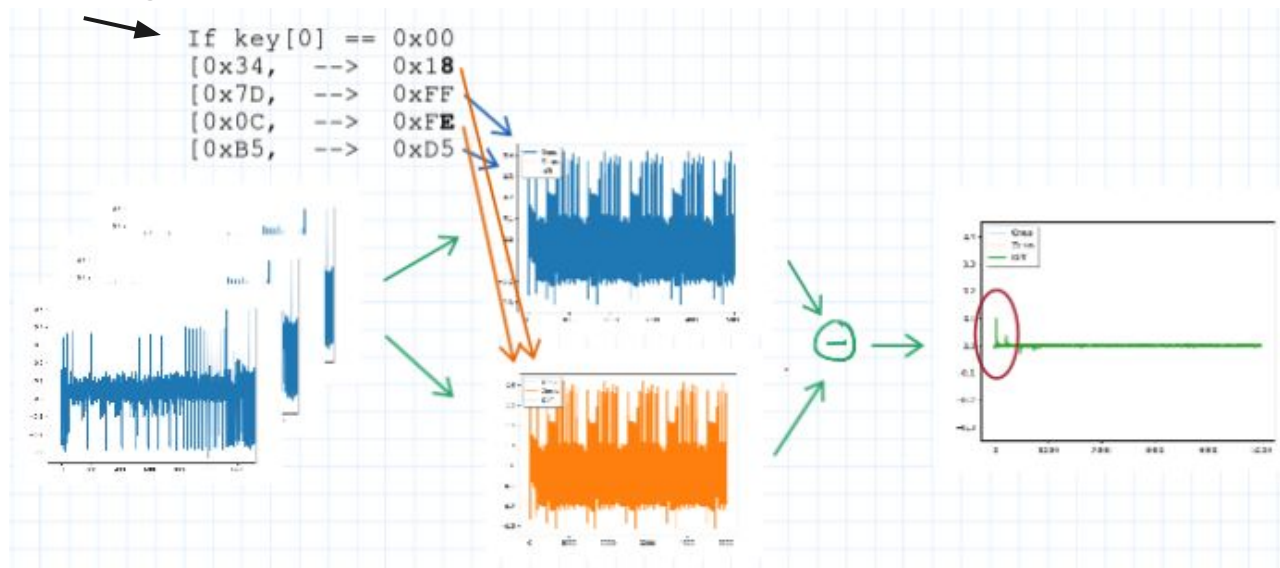
Step 1: Collect 1000s of power traces of device performing AES encryption





# Differential Power Analysis (DPA) attack on AES128

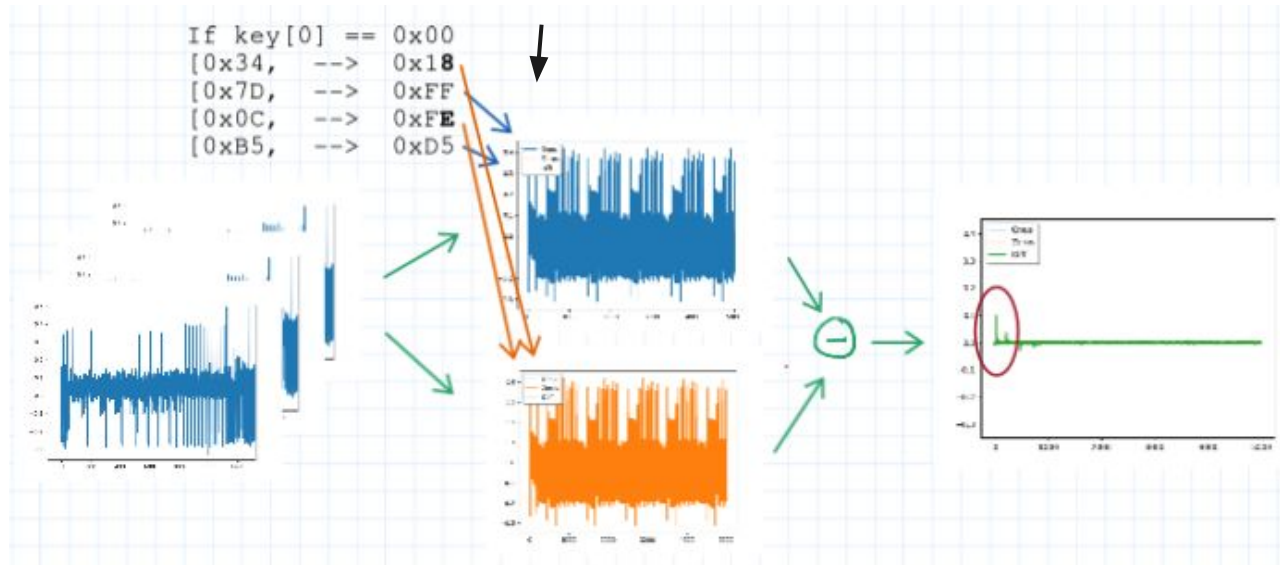
Step 2: Make a guess at key[0].





# Differential Power Analysis (DPA) attack on AES128

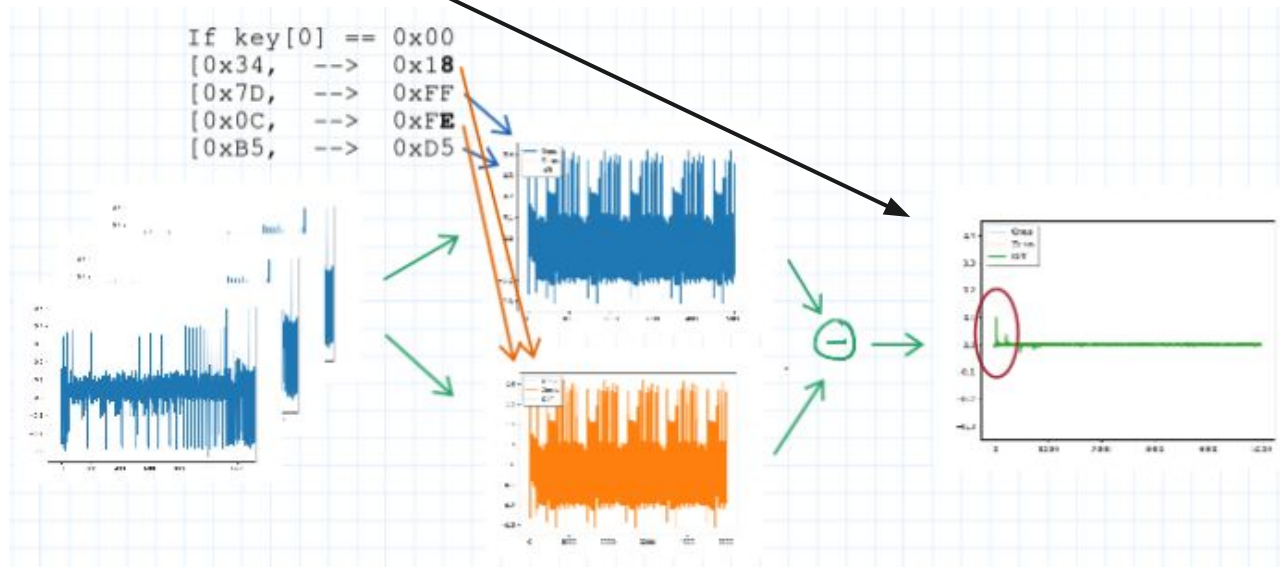
Step 3: Sort the power traces by `sbox[pt[0]^keyGuess]` and average the two groups.





# Differential Power Analysis (DPA) attack on AES128

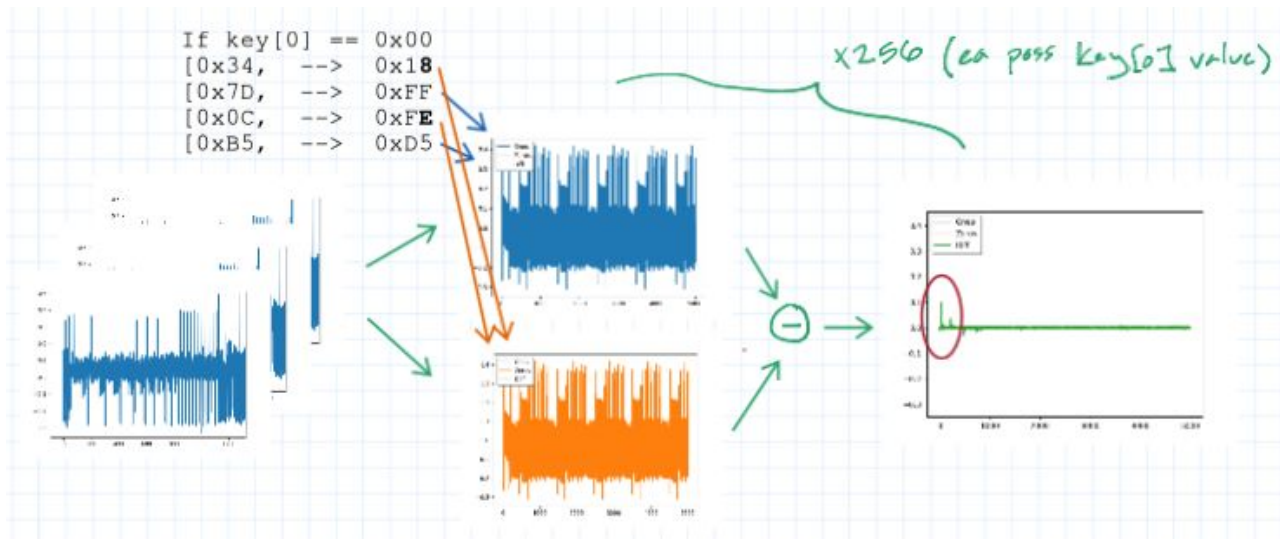
Step 4: Subtract the two groups.





# Differential Power Analysis (DPA) attack on AES128

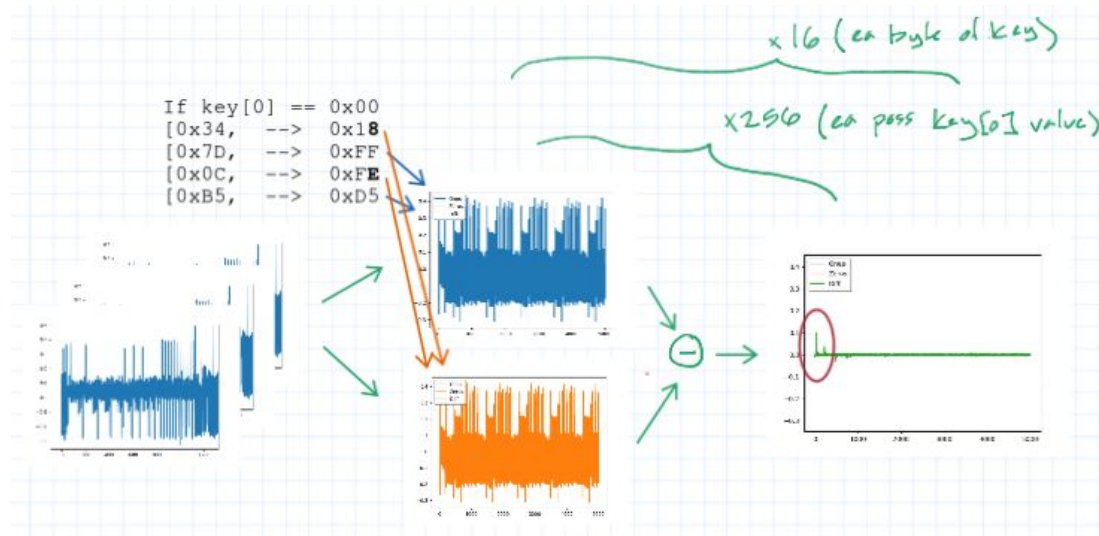
Step 5: Repeat #2-4 for each possible value of key[0] (0-255). The largest difference is the likeliest key!





# Differential Power Analysis (DPA) attack on AES128

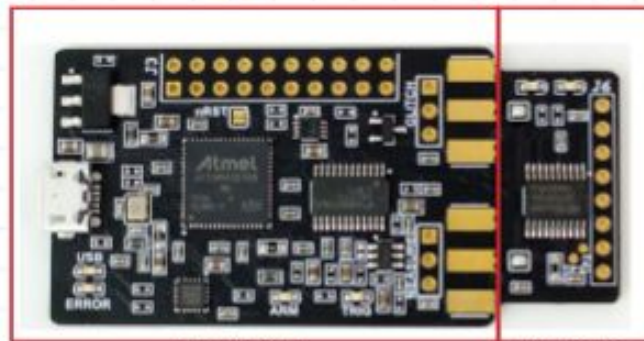
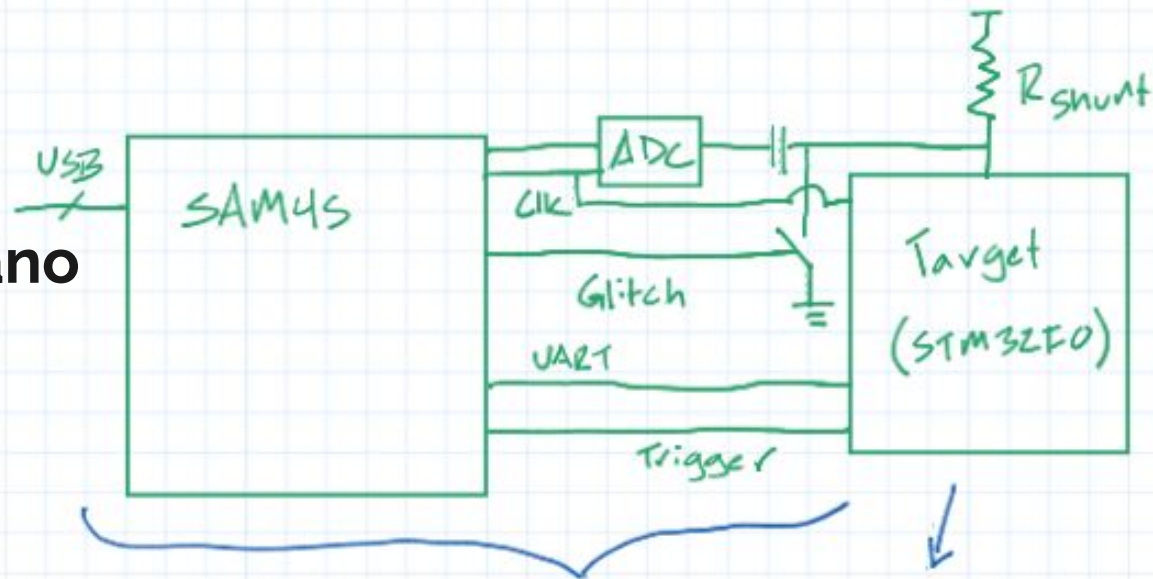
Step 6: Repeat #2-5 for each byte in the private key!



Slides:  
tinyurl.com  
/muujw9ze



# ChipWhisperer-Nano



CAPTURE Section

TARGET Section

Slides:  
tinyurl.com  
/muujw9ze



# DPA attack demonstration



Slides:  
tinyurl.com  
/muujw9ze



**Kahoot!**

Slides:  
tinyurl.com  
/muujw9ze

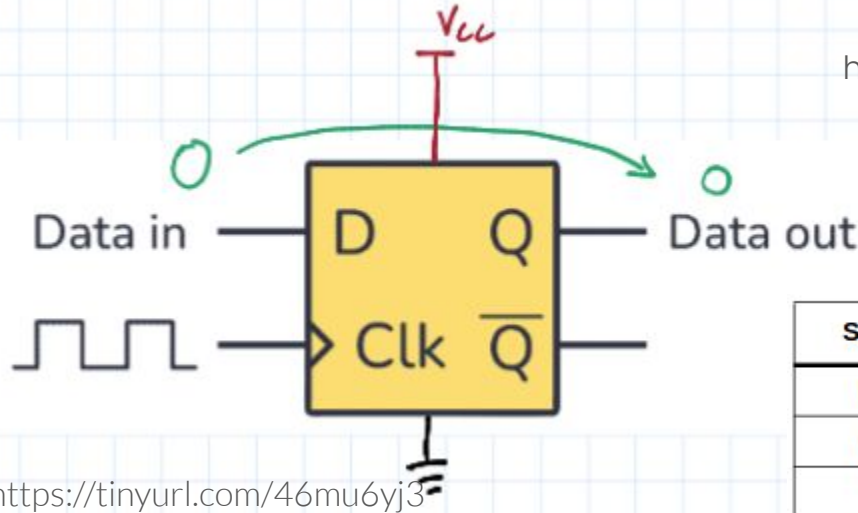


---

# Level 5: Fault injection attacks



# All components have required operating conditions



<https://tinyurl.com/46mu6yj3>

For the STM32F0:

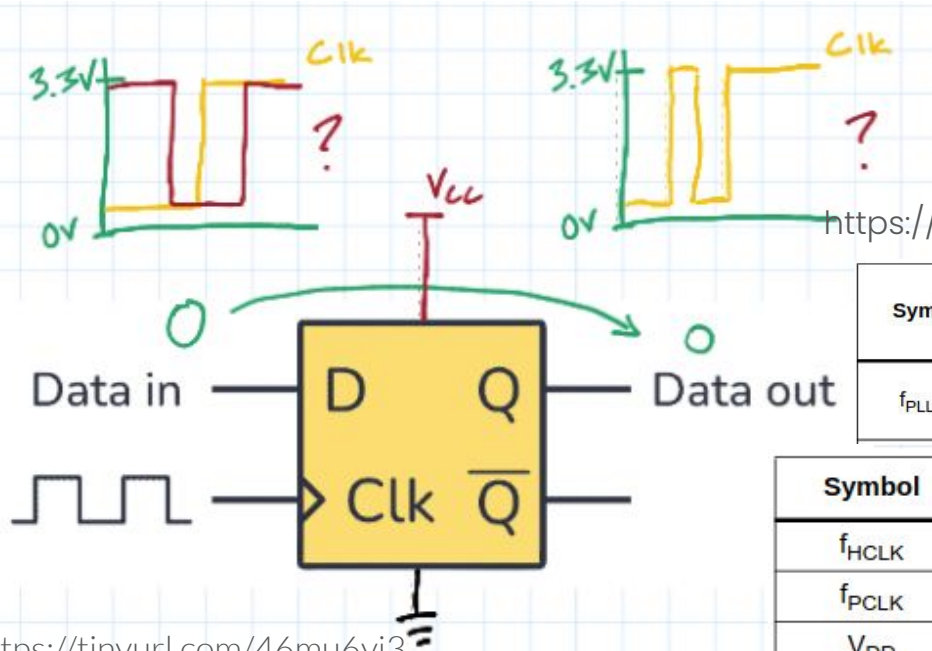
<https://www.st.com/resource/en/datasheet/stm32f042f4.pdf>

| Symbol               | Parameter                      | Value             |     |                   | Unit |
|----------------------|--------------------------------|-------------------|-----|-------------------|------|
|                      |                                | Min               | Typ | Max               |      |
| $f_{\text{PLL\_IN}}$ | PLL input clock <sup>(1)</sup> | 1 <sup>(2)</sup>  | 8.0 | 24 <sup>(2)</sup> | MHz  |
|                      | PLL input clock duty cycle     | 40 <sup>(2)</sup> | -   | 60 <sup>(2)</sup> | %    |

| Symbol            | Parameter                    | Min | Max | Unit |
|-------------------|------------------------------|-----|-----|------|
| $f_{\text{HCLK}}$ | Internal AHB clock frequency | 0   | 48  | MHz  |
| $f_{\text{PCLK}}$ | Internal APB clock frequency | 0   | 48  |      |
| $V_{\text{DD}}$   | Standard operating voltage   | 2.0 | 3.6 | V    |



## But what happens if these conditions aren't met?



<https://www.st.com/resource/en/datasheet/stm32f042f4.pdf>

For the STM32F0:

| Symbol        | Parameter                      | Value             |     |                   | Unit |
|---------------|--------------------------------|-------------------|-----|-------------------|------|
|               |                                | Min               | Typ | Max               |      |
| $f_{PLL\_IN}$ | PLL input clock <sup>(1)</sup> | 1 <sup>(2)</sup>  | 8.0 | 24 <sup>(2)</sup> | MHz  |
|               | PLL input clock duty cycle     | 40 <sup>(2)</sup> | -   | 60 <sup>(2)</sup> | %    |

| Symbol     | Parameter                    | Min | Max | Unit |
|------------|------------------------------|-----|-----|------|
| $f_{HCLK}$ | Internal AHB clock frequency | 0   | 48  | MHz  |
| $f_{PCLK}$ | Internal APB clock frequency | 0   | 48  |      |
| $V_{DD}$   | Standard operating voltage   | 2.0 | 3.6 | V    |



## But what happens if these conditions aren't met?



Data:



We don't know! Most likely reset, but occasionally:

- Skipped instructions
- Altered/corrupted instructions
- Altered/corrupted memory

For the STM32F0:  
stm32f042f4.pdf

|                   | Unit |
|-------------------|------|
| 24 <sup>(2)</sup> | MHz  |
| 60 <sup>(2)</sup> | %    |

|           | Max                        | Unit |
|-----------|----------------------------|------|
|           | 0                          | 48   |
|           | 0                          | 48   |
| frequency | 2.0                        | 3.6  |
| VDD       | Standard operating voltage | V    |

Slides:  
tinyurl.com  
/muujw9ze



## Fault injection attack on password check

```
#if SS_VER == SS_VER_2_1
uint8_t password(uint8_t cmd, uint8_t scmd, uint8_t len, uint8_t* pw)
#else
uint8_t password(uint8_t* pw, uint8_t len)
#endif
{
    char passwd[] = "touch";
    char passok = 1;
    int cnt;

    trigger_high();

    //Simple test - doesn't check for too-long password!
    for(cnt = 0; cnt < 5; cnt++){
        if (pw[cnt] != passwd[cnt]){
            passok = 0;
        }
    }

    trigger_low();

    if(passok) access = true;

    simpleserial_put('r', 1, (uint8_t*)&passok);
    return 0x00;
}
```

Slides:  
tinyurl.com  
/muujw9ze



## Fault injection attack on password check

```
#if SS_VER == SS_VER_2_1
uint8_t password(uint8_t cmd, uint8_t scmd, uint8_t len, uint8_t* pw)
#else
uint8_t password(uint8_t* pw, uint8_t len)
#endif
{
    char passwd[] = "touch";
    char passok = 1;
    int cnt;

    trigger_high();

    //Simple test - doesn't check for too-long password!
    for(cnt = 0; cnt < 5; cnt++){
        if (pw[cnt] != passwd[cnt]){
            passok = 0;
        }
    }

    trigger_low();

    if(passok) access = true;

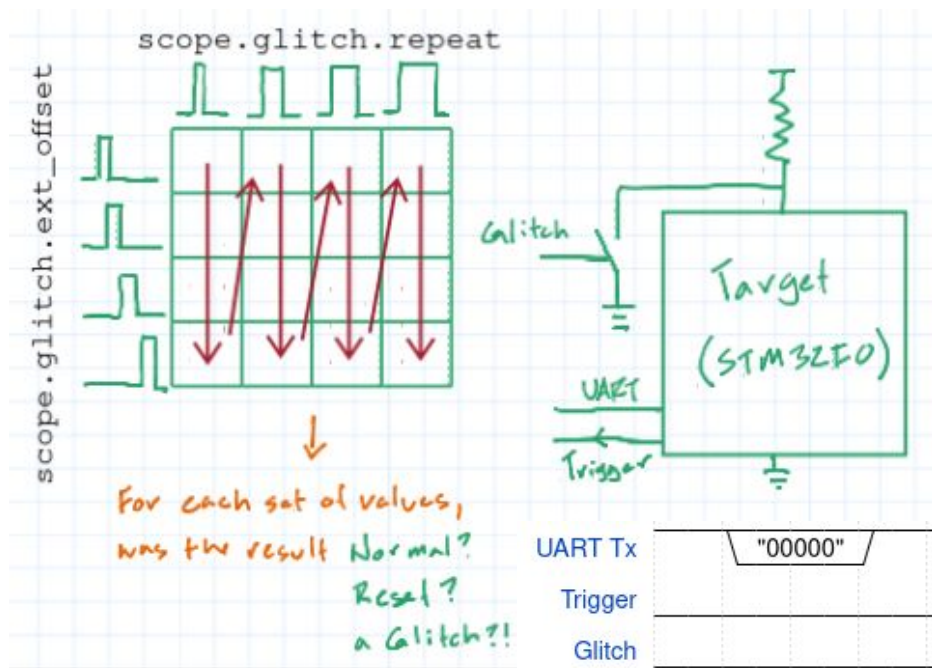
    simpleserial_put('r', 1, (uint8_t*)&passok);
    return 0x00;
}
```

```
93:simpleserial-glitch.c **** //Simple test - doesn't check for
94:simpleserial-glitch.c **** for(cnt = 0; cnt < 5; cnt++){
360 .loc 1 94 5 view .LVU75
361 .loc 1 94 13 is_stmt 0 view .LVU76
    , #0
    , #20]
    5 view .LVU77

366 003e 11E0 b .L16
367 .L18:
95:simpleserial-glitch.c **** if (pw[cnt] != passwd[cnt]){
368 .loc 1 95 9 is_stmt 1 view .LVU78
369 .loc 1 95 15 is_stmt 0 view .LVU79
370 0040 7B69 ldr r3, [r7, #20]
371 .LVL28:
372 .loc 1 95 15 view .LVU80
373 0042 3A68 ldr r2, [r7]
374 0044 D318 adds r3, r2, r3
375 0046 1A78 ldrb r2, [r3]
376 .loc 1 95 30 view .LVU81
377 0048 0C23 movs r3, #12
378 004a F918 adds r1, r7, r3
379 004c 7B69 ldr r3, [r7, #20]
380 004e CB18 adds r3, r1, r3
381 0050 1B78 ldrb r3, [r3]
382 .loc 1 95 12 view .LVU82
383 0052 9A42 cmp r2, r3
384 0054 03D0 beq .L17
96:simpleserial-glitch.c **** passok = 0;
385 .loc 1 96 13 is_stmt 1 view .LVU83
386 .loc 1 96 20 is_stmt 0 view .LVU84
387 0056 0B23 movs r3, #11
388 0058 FB18 adds r3, r7, r3
389 005a 0022 movs r2, #0
390 005c 1A70 strb r2, [r3]
```



## Fault injection attack on password check



```
trigger_high();  
  
//Simple test - doesn't check for too-long password!  
for(cnt = 0; cnt < 5; cnt++){  
    if (pw[cnt] != passwd[cnt]){  
        passok = 0;  
    }  
}
```

Can we corrupt this  
instruction???

UART Tx

Trigger

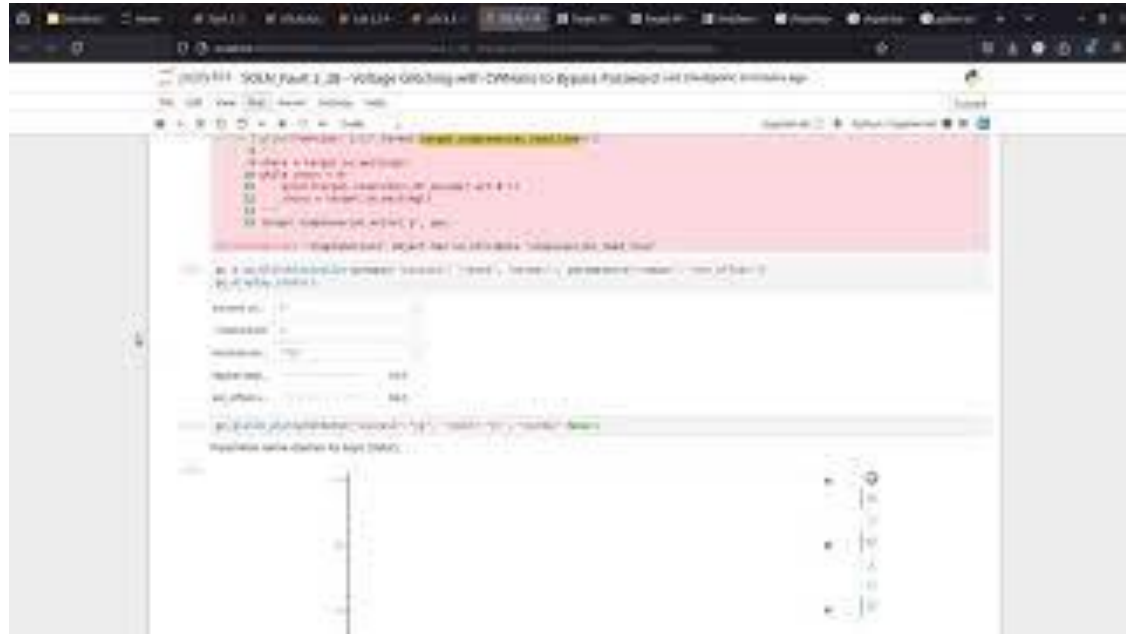
Glitch

"00000"

Slides:  
tinyurl.com  
/muujw9ze



# Fault injection attack on password check



Slides:  
tinyurl.com  
/muujw9ze

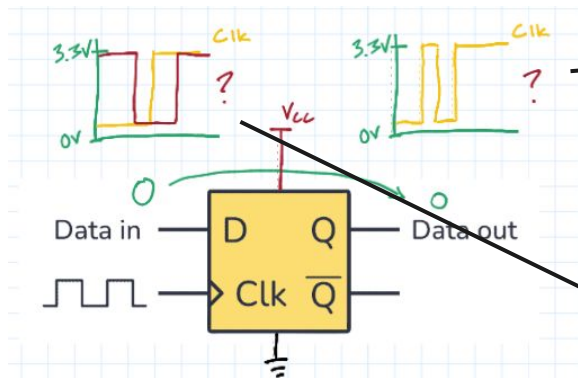


**A fault doesn't need to make sense to work, it just needs to be repeatable.**

---

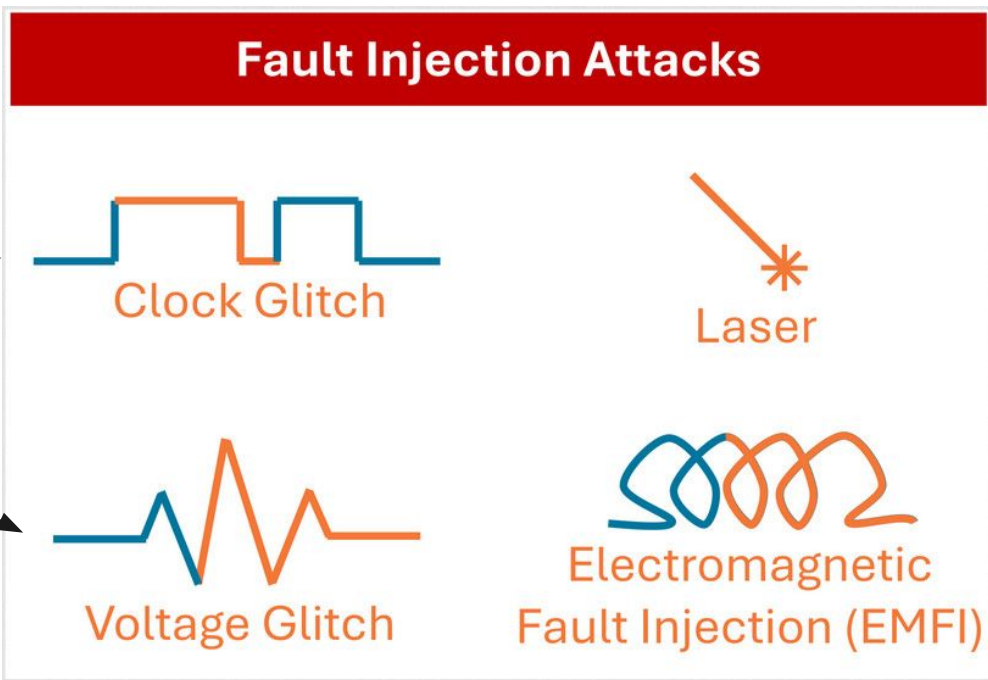


## Ways to induce faults



<https://tinyurl.com/46mu6yj3>

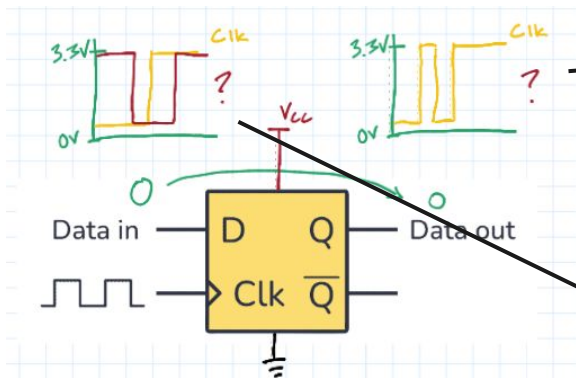
<https://tinyurl.com/4u434mc2>



Slides:  
tinyurl.com  
/muujw9ze

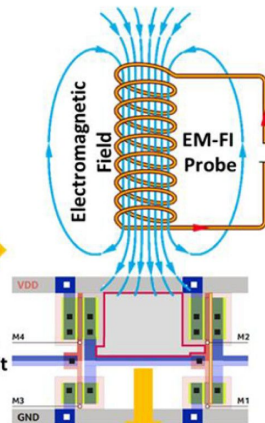
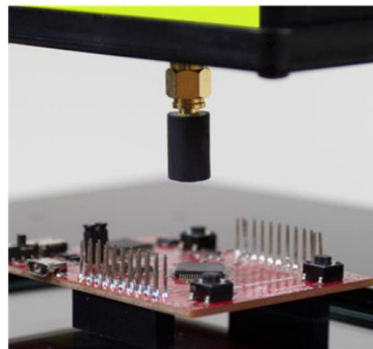


## Ways to induce faults



<https://tinyurl.com/46mu6yj3>

<https://tinyurl.com/4u434mc2>



Eddy Current → Transistor Error

<https://tinyurl.com/sn4mravy>

Laser

Electromagnetic  
Fault Injection (EMFI)

Slides:  
tinyurl.com  
/muujw9ze



# Fault injection attack on LPC code read protection



<https://tinyurl.com/yvehn3dx>

[https://os.mbed.com/media/cache/platforms/LPC1768.jpg.250x250\\_q85.jpg](https://os.mbed.com/media/cache/platforms/LPC1768.jpg.250x250_q85.jpg)

Table 372. Code Read Protection options

| Name   | Pattern programmed in 0x0000 02FC | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NO_ISP | 0x4E69 7370                       | Prevents sampling of pin PIO0_1 for entering ISP mode. PIO0_1 is available for other uses.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| CRP1   | 0x12345678                        | <p>Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions:</p> <ul style="list-style-type: none"><li>• Write to RAM command should not access RAM below 0x1000 0300. Access to addresses below 0x1000 0200 is disabled.</li><li>• Copy RAM to flash command can not write to Sector 0.</li><li>• Erase command can erase Sector 0 only when all sectors are selected for erase.</li><li>• Compare command is disabled.</li><li>• Read Memory command is disabled.</li></ul> <p>This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the flash.</p> |
| CRP2   | 0x87654321                        | <p>Access to chip via the SWD pins is disabled. The following ISP commands are disabled:</p> <ul style="list-style-type: none"><li>• Read Memory</li><li>• Write to RAM</li><li>• Go</li><li>• Copy RAM to flash</li><li>• Compare</li></ul> <p>When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| CRP3   | 0x43218765                        | Access to chip via the SWD pins is disabled. ISP entry by pulling PIO0_1 LOW is disabled if a valid user code is present in flash sector n                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

Slides:  
tinyurl.com  
/muujw9ze



# Fault injection attack on LPC code read protection (CRP)



<https://tinyurl.com/yvehn3dx>

[https://os.mbed.com/media/cache/platforms/LPC1768.jpg.250x250\\_q85.jpg](https://os.mbed.com/media/cache/platforms/LPC1768.jpg.250x250_q85.jpg)

Table 372. Code Read Protection options

| Name   | Pattern programmed in 0x0000 02FC | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Value in CRP reg | Meaning       |
|--------|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|---------------|
| NO_ISP | 0x4E69 7370                       | Prevents sampling of pin PIO0_1 for code read protection available for other uses.                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 0x12345678       | CRP1          |
| CRP1   | 0x12345678                        | Access to chip via the SWD pins is disabled. Flash update using the following ISP commands: <ul style="list-style-type: none"><li>Write to RAM command should not be used.</li><li>Copy RAM to flash command can be used.</li><li>Erase command can erase Sector selected for erase.</li><li>Compare command is disabled.</li><li>Read Memory command is disabled.</li></ul> This mode is useful when CRP is required but all sectors can not be erased. This mode is disabled in case of partial updates to implement checksum mechanism to verify code. | 0x87654321       | CRP2          |
| CRP2   | 0x87654321                        | Access to chip via the SWD pins is disabled. All commands are disabled: <ul style="list-style-type: none"><li>Read Memory</li><li>Write to RAM</li><li>Go</li><li>Copy RAM to flash</li><li>Compare</li></ul> When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.                                                                                                                                                                                                                                                         | 0x43218765       | CRP3          |
| CRP3   | 0x43218765                        | Access to chip via the SWD pins is disabled. ISP entry by pulling PIO0_1 LOW is disabled if a valid user code is present in flash sector n.                                                                                                                                                                                                                                                                                                                                                                                                               | 0x4E697370       | NO_ISP        |
|        |                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Any other value  | No protection |



# Fault injection attack on LPC code read protection

## Useful tools and links:

lpc21isp - Software to program LPC-family microcontrollers using ISP  
<http://elecing.dti.nl/frank/arm/BareMetal/LPC612/index.html>  
[http://elecing.dti.nl/frank/arm/BareMetal/LPC612/serial\\_target](http://elecing.dti.nl/frank/arm/BareMetal/LPC612/serial_target)  
<http://elecing.dti.nl/frank/arm/BareMetal/LPC1114/index.html>  
<http://gnudd.com/cortex-m3/gt> <http://www.gnudd.com/wdcortex-m3.pdf>  
<http://www.stm32.com/Product/ARM/NXP/LPC-P2118/>  
[http://www.bauing.uni-kl.de/wr\\_projects/arm\\_projects/lpc2k\\_bundle\\_posi/lpc213x\\_lpc214x\\_examples\\_20061206.zip](http://www.bauing.uni-kl.de/wr_projects/arm_projects/lpc2k_bundle_posi/lpc213x_lpc214x_examples_20061206.zip)



```
// Bootloader
```

```
...
```

```
#define CRP1 0x12345678
```

```
#define CRP2 0x87654321
```

```
uint32_t crp = *CRP_REG;
```

```
if(crp == CRP1){...}
```

```
else if(crp == CRP2){...}
```

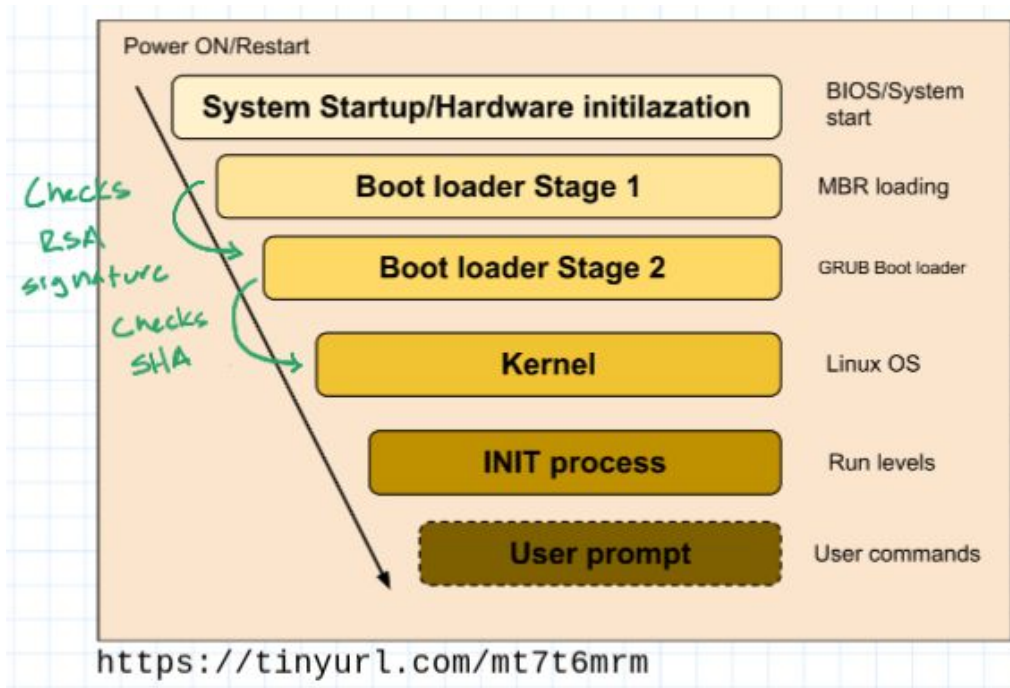
Glitch here!



## Fault injection attack on Xbox 360 (2011)



<https://tinyurl.com/44ywmn6a>

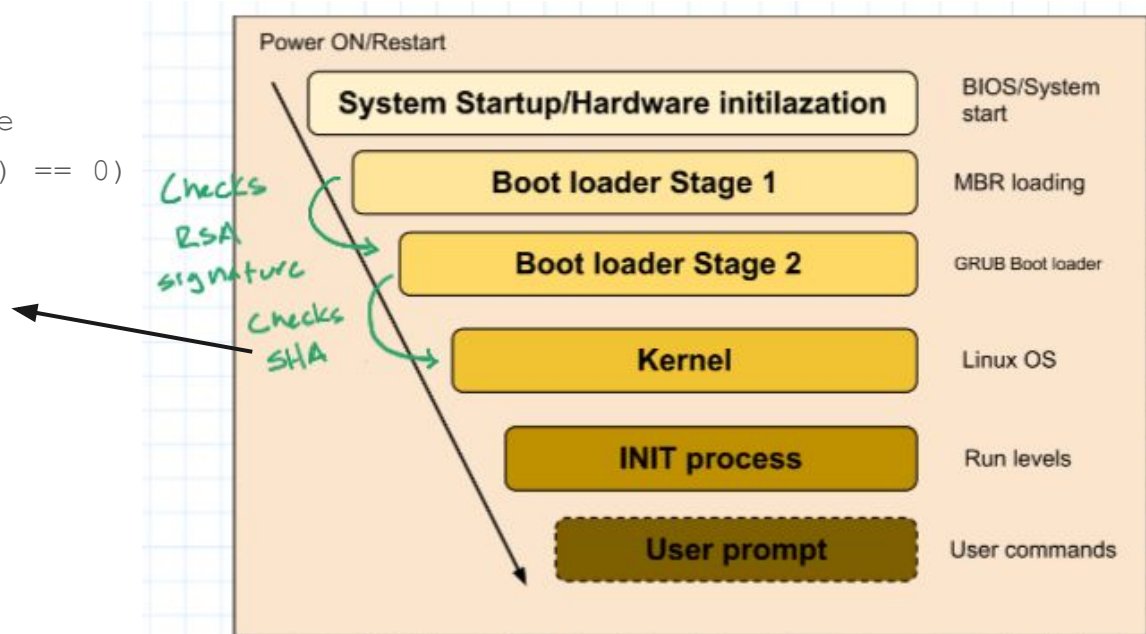


Slides:  
tinyurl.com  
/muujw9ze



## Fault injection attack on Xbox 360 (2011)

```
// Overly simplified Xbox code
if(memcmp(ker_sha, exp_sha, n) == 0)
{
    run_kernel();
}
else error();
```



<https://tinyurl.com/mt7t6mrm>

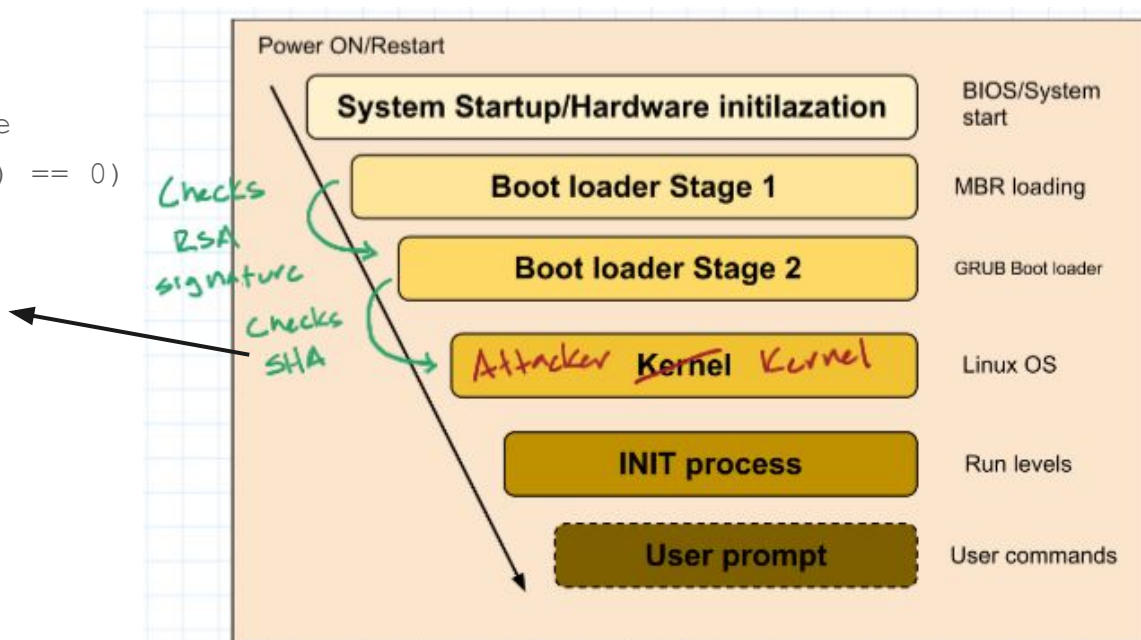
Slides:  
tinyurl.com  
/muujw9ze



## Fault injection attack on Xbox 360 (2011)

```
// Overly simplified Xbox code
if(memcmp(ker_sha, exp_sha, n) == 0)
{
    run_kernel();
}
else error();
```

Glitch this!



<https://tinyurl.com/mt7t6mrm>

Slides:  
tinyurl.com  
/muujw9ze



**Discuss!**

How would you fix the code?

```
// Overly simplified Xbox code
if(memcmp(ker_sha, exp_sha, n) == 0)
{
    run_kernel();
}
else error();
```

Slides:  
tinyurl.com  
/muujw9ze



**Kahoot!**

Slides:  
tinyurl.com  
/muujw9ze



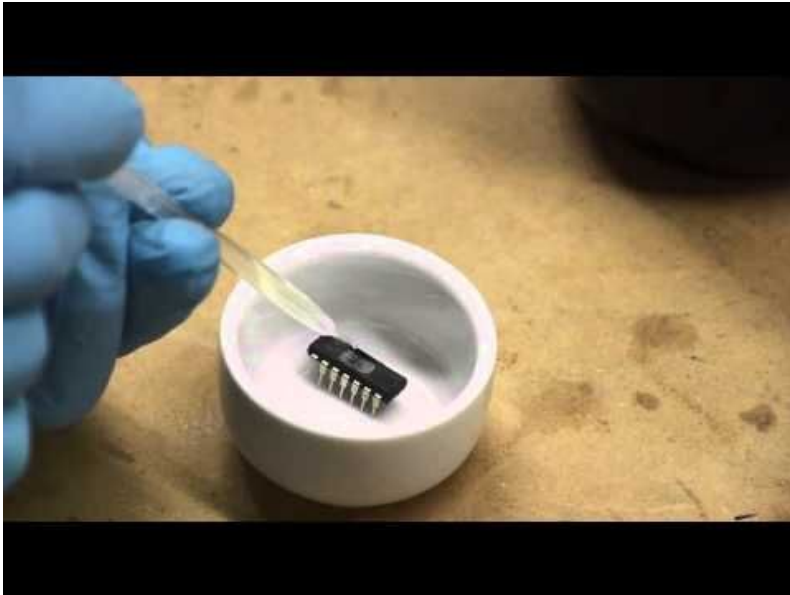
---

# Level 6: IC decapping

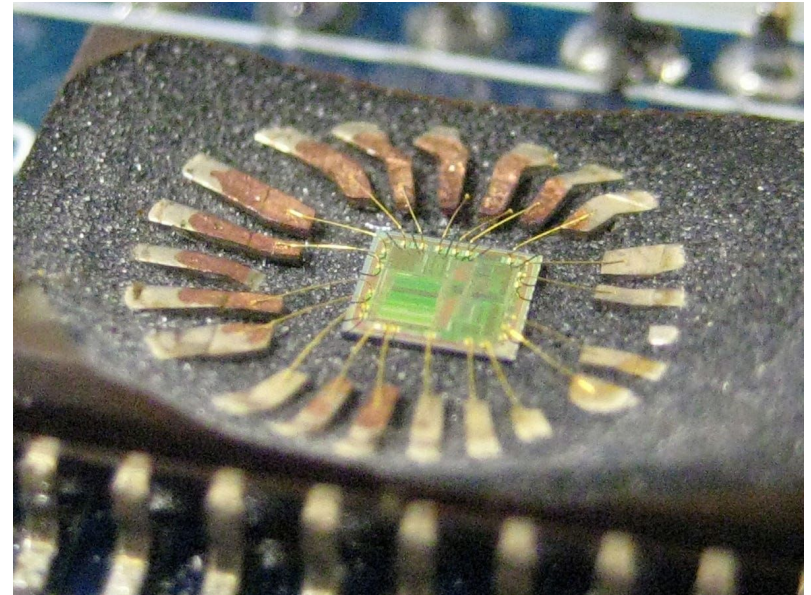
Slides:  
tinyurl.com  
/muujw9ze



## Exposes the internal silicon



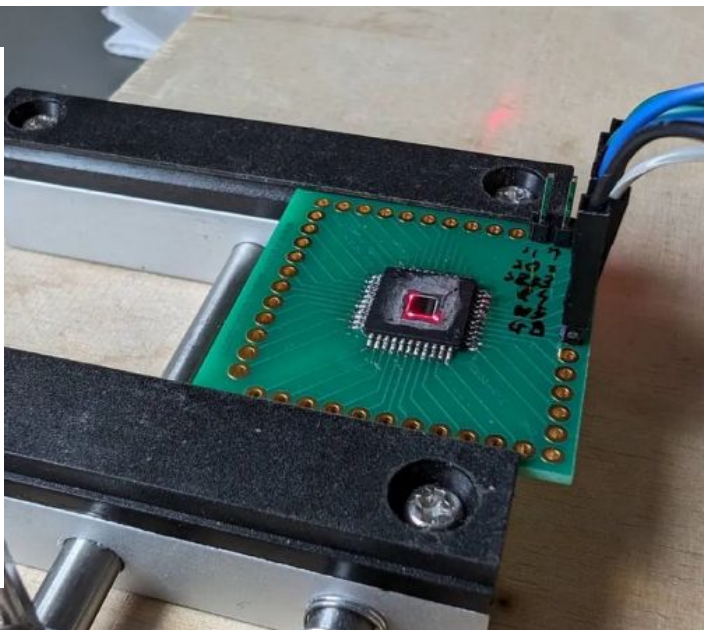
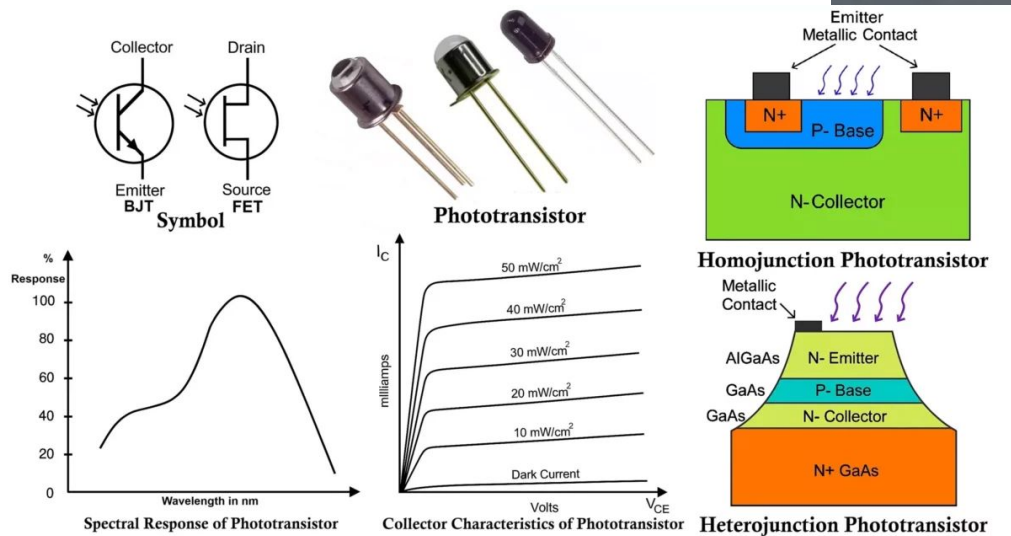
<https://i.ytimg.com/vi/mTIFStxAVz4/hqdefault.jpg>



<https://tinyurl.com/bdz33ttz>



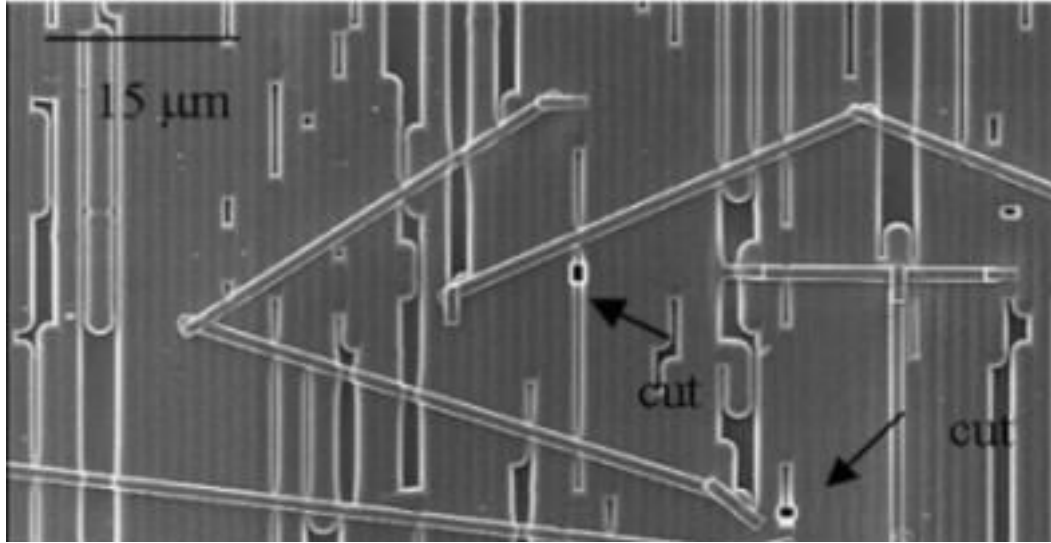
# Optical side-channel / fault injection attacks



Slides:  
tinyurl.com  
/muujw9ze

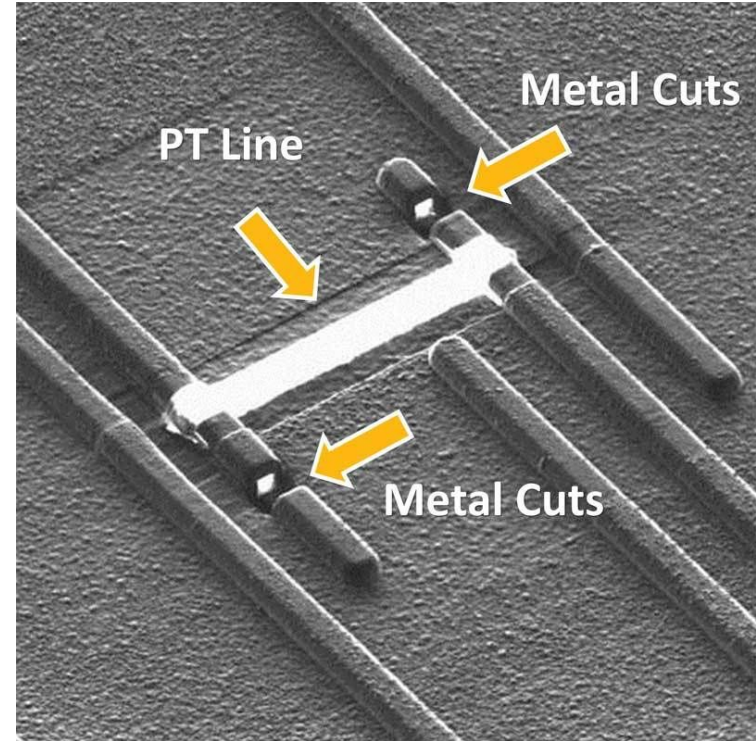


## Focused-ion beam (FIB) editing



<https://tinyurl.com/bdahjcw5>

<https://tinyurl.com/yhzw5btj>



Slides:  
tinyurl.com  
/muujw9ze



---

# Countermeasures



# Fighting side-channel attacks

## Noncorrelating/Constant Time

- Random delays  
`gets(enteredPw);`  
`// random delay`  
`if(strcmp(enteredPw, correctPw) ==`  
`0)...`
- Constant-time memcmp  
`// diff = 0`  
`// n times:`  
`// diff = diff | (c1[n] ^ c2[n])`  
`// return diff (0 if c1 == c2)`

## Noncorrelating/Constant Power Consumption

- Use more of the IC when conducting sensitive operations
- Use side-channel-resistant cryptographic libraries (masking, blinding)



# Fighting fault injection attacks

## Nontrivial constants

- `#define FALSE 0x12345678`  
`#define TRUE 0x87654321`
- `memcmp`  
`#define C1 0xA0B0C000`  
`#define C2 0x01020300`  
`// diff = 0`  
`// n times:`  
`// tmp = (C1|c1[n]) ^ (C2|c2[n])`  
`// diff = diff | tmp`  
`// return diff`  
`// (0xA1B2C300 if c1 == c2)`

## Control flow verification

- Check that loops end with the right count
- Double-check critical operations  
`if(memcmp(in, actual, n)==0)`  
`{`  
 `if(memcmp(in, actual, n)==0)`  
 `{`  
 `// Do stuff`  
 `}`  
`}`

Slides:  
tinyurl.com  
/muujw9ze



Determine *what* secrets you're trying to protect, *who* could attack them, and *how*. Then employ countermeasures against those *specific* attacks.

---

Slides:  
tinyurl.com  
/muujw9ze



---

# Conclusion

Slides:  
tinyurl.com  
/muujw9ze



**Discuss!**

Summarize the most important points of the last hour with the person next to you.

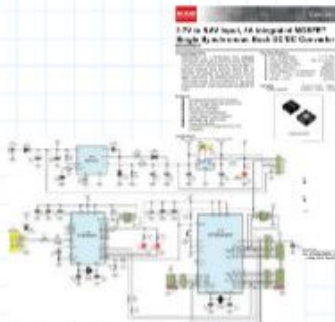


## Closing thoughts

Level 1:  
Researching



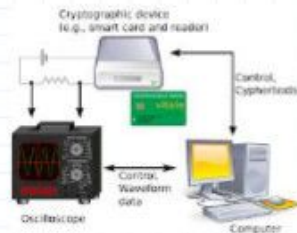
Level 2:  
Snooping



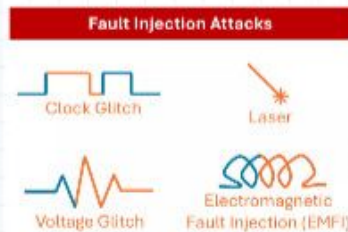
Level 3:  
PCB  
modifications



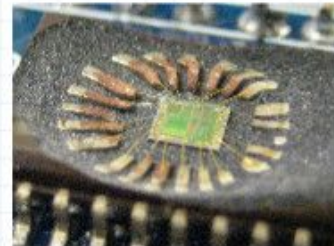
Level 4:  
Side-channel  
attacks



Level 5:  
Fault-injection  
attacks



Level 6:  
IC decapping



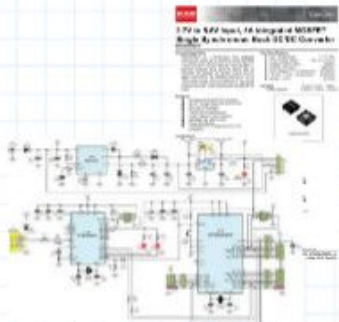


## Questions?

Level 1:  
Researching



Level 2:  
Snooping

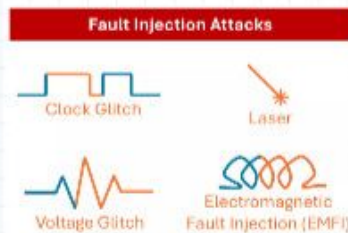


Level 3:  
PCB  
modifications

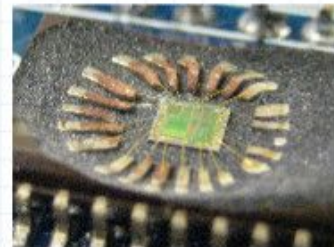


Level 4:  
Side-channel  
attacks

Level 5:  
Fault-injection  
attacks



Level 6:  
IC decapping



Slides:  
tinyurl.com  
/muujw9ze



---

# Thank you!

Slides:  
tinyurl.com  
/muujw9ze



**Kahoot!**