

**XIII
JORNADAS
STIC
CCN-CERT**



**Los secretos de tu
chip IoT**

**COMUNIDAD Y CONFIANZA,
BASES DE NUESTRA CIBERSEGURIDAD**

#XIIIJORNADASCCNCERT





Javier Tallón – Technical Director



 jtallon@jtsec.es

 [@javiertallon](https://twitter.com/javiertallon) / [@jtsecES](https://twitter.com/jtsecES)

Índice

1. Introducción

1. Vulnerabilidades del IoT

2. Accediendo al firmware

3. Técnicas de ataque a la protección contra lectura

1. *Heart of Darkness*

2. *UV-C Security Fuse Erase*

3. *Coold Boot Stepping*

4. *Flash Lock Race*

5. Inyección de Faltas

6. *Side Channel Attack*

4. Conclusiones

1. Introducción

Predicción del Crecimiento IoT

Connected devices (billions)



	2016	2022	CAGR
Wide-area IoT	0.4	2.1	30%
Short-range IoT	5.2	16	20%
PC/laptop/tablet	1.6	1.7	0%
Mobile phones	7.3	8.6	3%
Fixed phones	1.4	1.3	0%
	16 billion	29 billion	10%

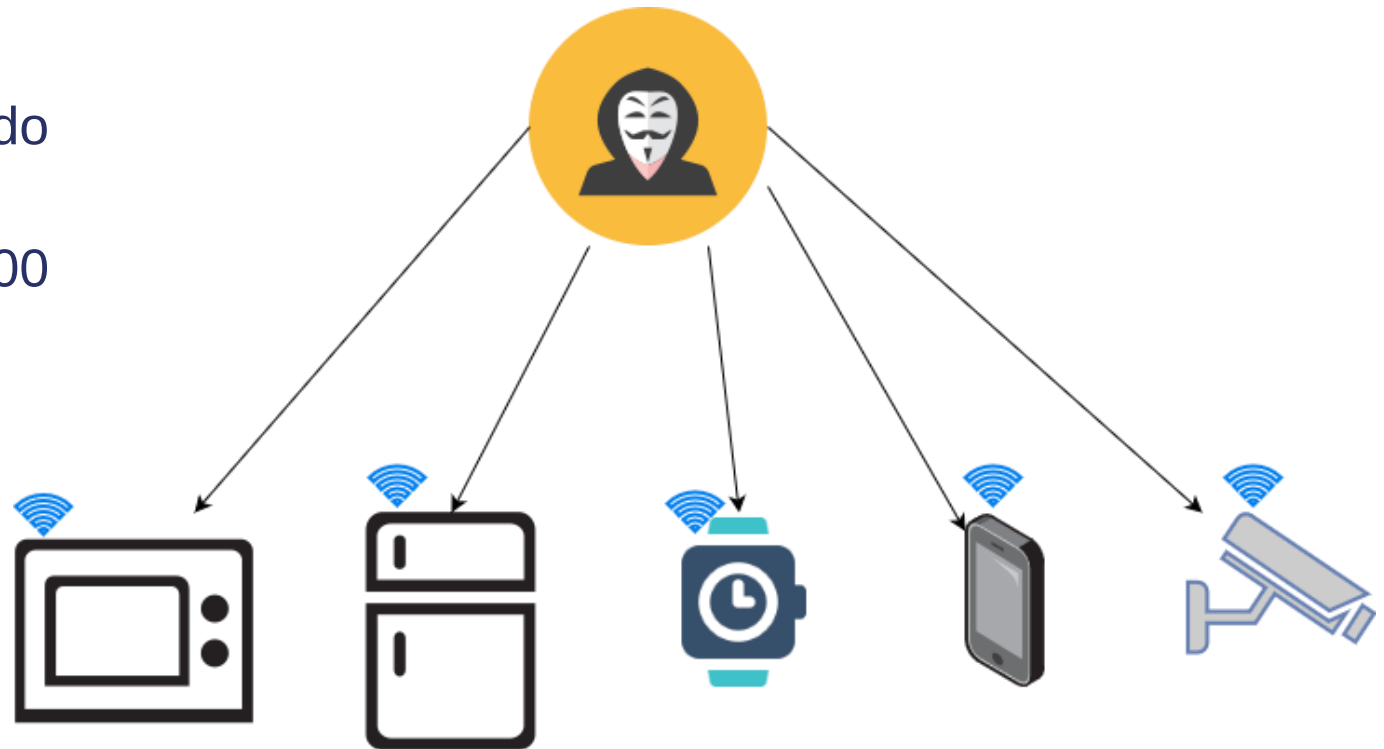
Fuente: <https://www.ericsson.com/en/mobility-report/internet-of-things-forecast>

1. Introducción

IoT en el Punto de Mira de los Hackers

- IoT *botnets*
- Denegación de Servicio Distribuido (**DDOS**)
- Okiru y Masuta: Más de 700,000 dispositivos infectados

Fuente: <https://krebsonsecurity.com/2019/09/satori-iot-botnet-operator-pleads-guilty/>



1. Introducción

Vulnerabilidades IoT. *eHealth*

Security

Hacking these medical pumps is as easy as copying a booby-trapped file over the network

Uncle Sam sounds alarm after Windows CE SMB left wide open on hospital equipment

By Thomas Claburn in San Francisco 13 Jun 2019 at 19:22 34  SHARE ▼



ÚLTIMA HORA 19/11/2019 16:47

Publicado un informe de código dañino sobre Raccoon Stealer

[Inicio](#) > [Seguridad al día](#) > [Noticias de actualidad](#) > [Cómo implementar servicios de autenticación de doble factor](#)

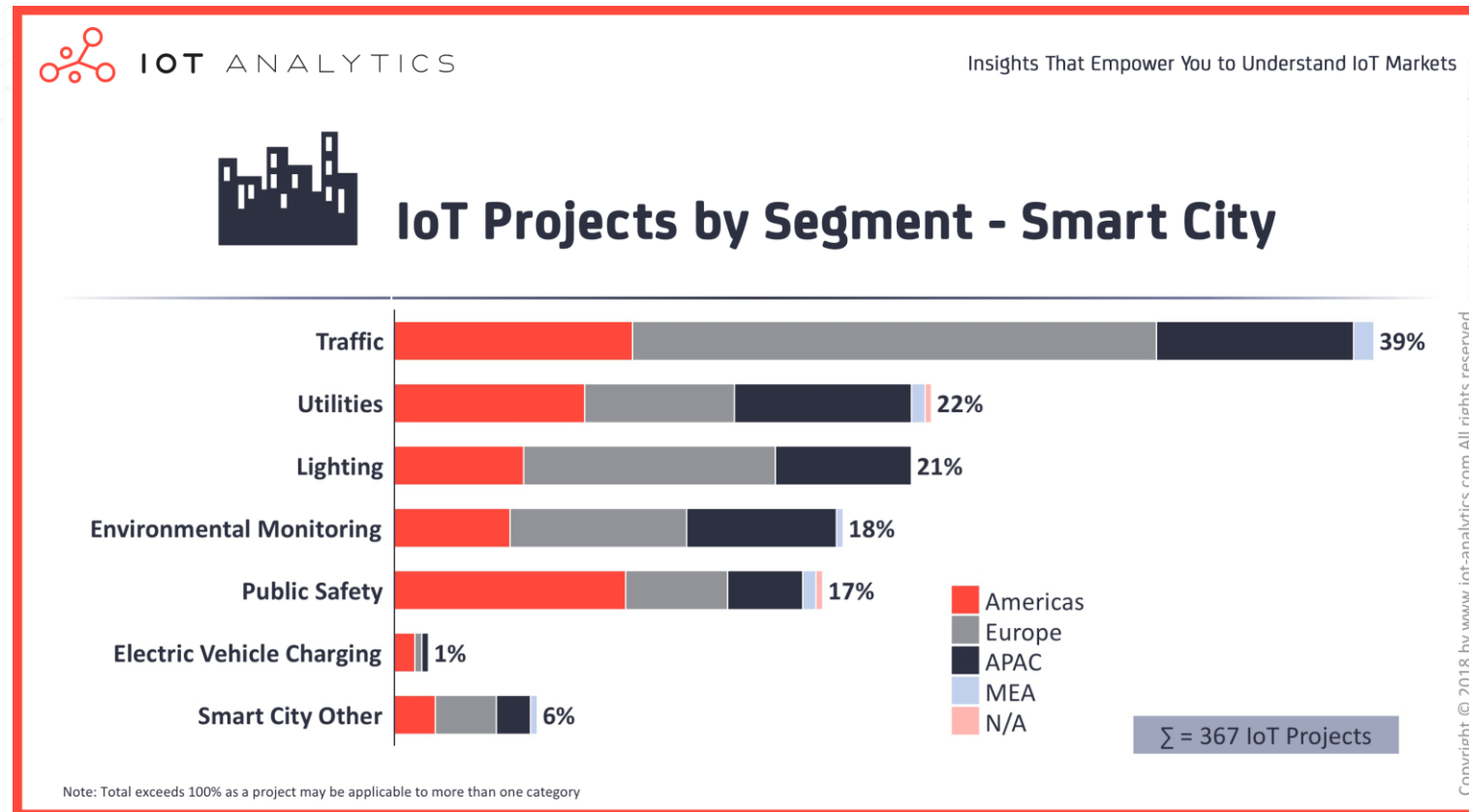
Las bombas de insulina de Medtronic anteriores a 2013 podrían sufrir ciberataques

Fuente: <https://www.ccn-cert.cni.es/seguridad-al-dia/noticias-seguridad/8342-las-bombas-de-insulina-de-medtronic-antiores-a-2013-podrian-sufrir-ciberataques.html>

Fuente: https://www.theregister.co.uk/2019/06/13/medical_workstation_vulnerabilities/

1. Introducción

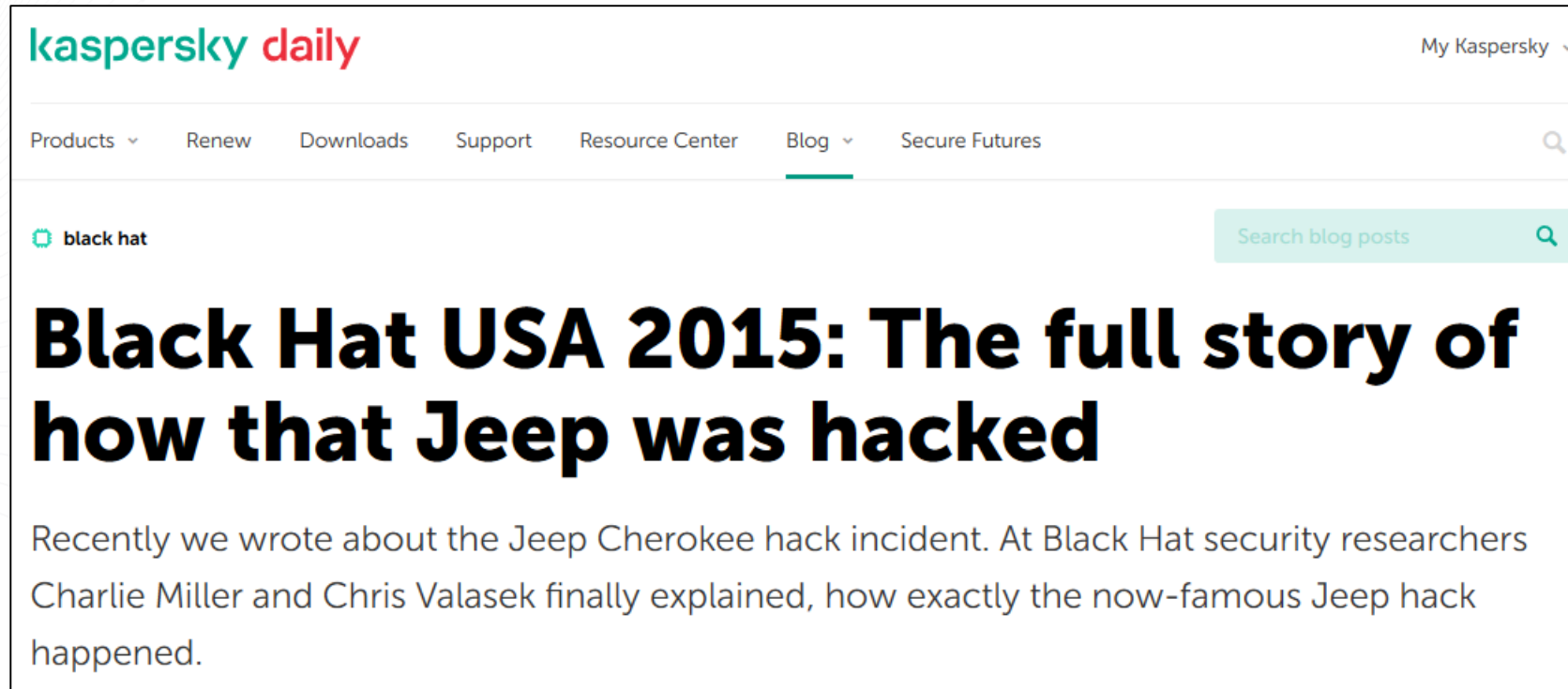
Vulnerabilidades IoT. *Smart Cities*



Fuente: <https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/>

1. Introducción

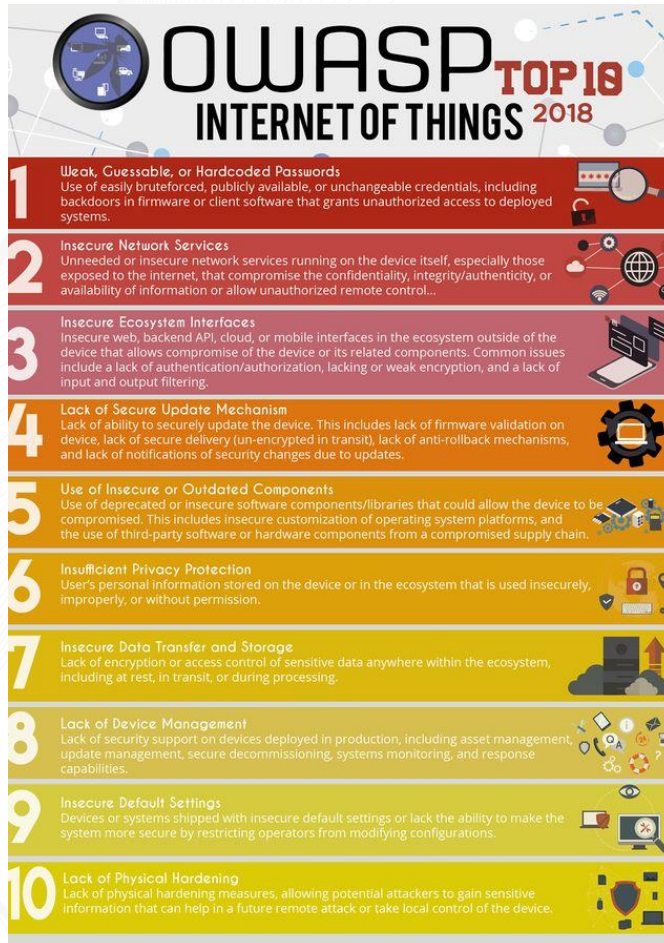
Vulnerabilidades IoT. *Smart Cities*



Fuente: <https://www.kaspersky.com/blog/blackhat-jeep-cherokee-hack-explained/9493/>

1. Introducción

Vulnerabilidades IoT

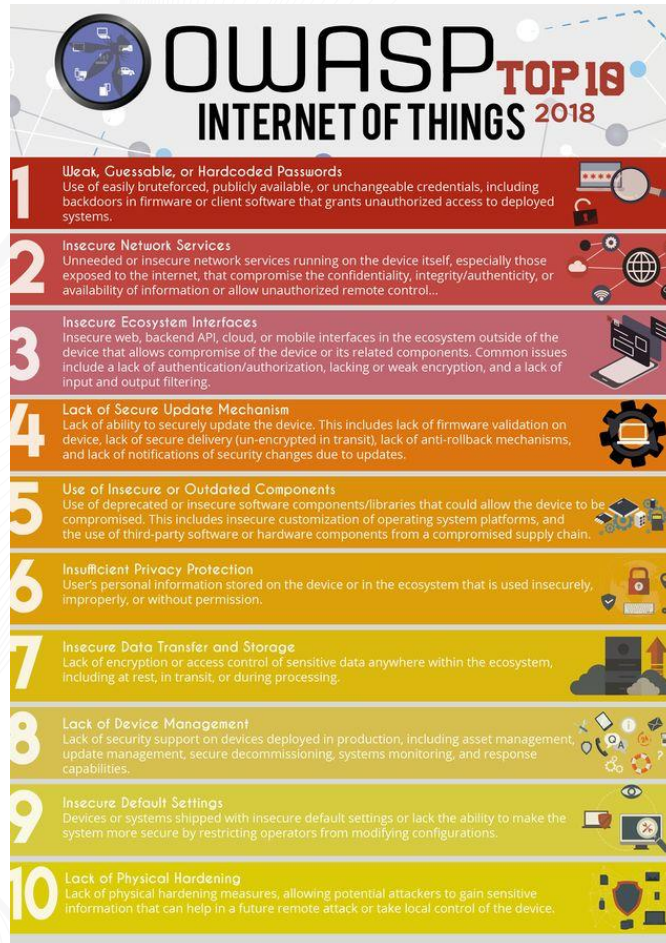


1. Contraseñas débiles, inseguras o por defecto.
2. Servicios expuestos a Internet inseguros
3. Ecosistema inseguro (cifrado débil, falta de autenticación, integridad, etc.)
4. Falta de mecanismos de actualización seguros
5. Uso de componentes desactualizados
6. Protección de privacidad insuficiente
7. Transferencia y almacenamiento de datos de forma insegura
8. Monitorización insuficiente
9. Configuración por defecto insegura
10. Falta de seguridad física

Fuente: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

1. Introducción

Vulnerabilidades IoT



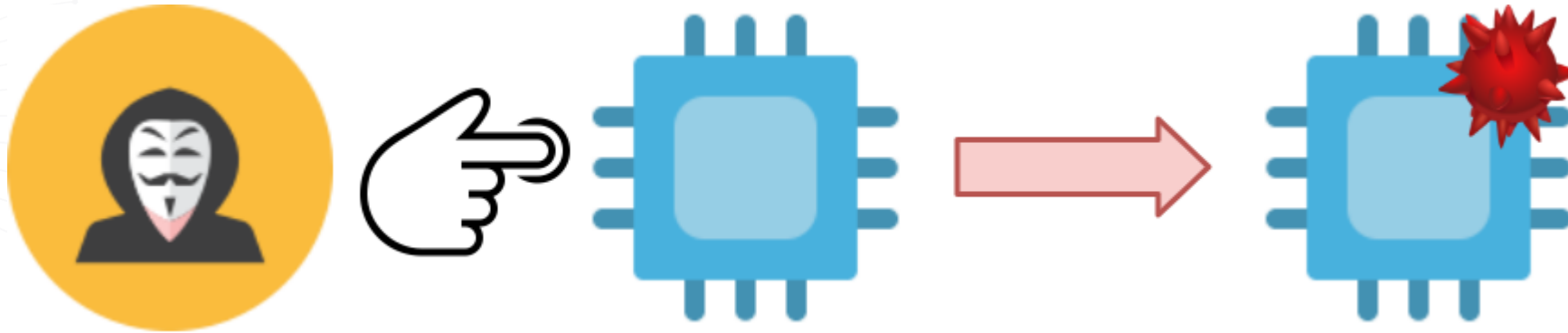
1. Contraseñas débiles, inseguras o por defecto.
2. Servicios expuestos a Internet inseguros
3. Ecosistema inseguro (cifrado débil, falta de autenticación, integridad, etc.)
4. Falta de mecanismos de actualización seguros
5. Uso de componentes desactualizados
6. Protección de privacidad insuficiente
7. Transferencia y almacenamiento de datos de forma insegura
8. Monitorización insuficiente
9. Configuración por defecto insegura
10. Falta de seguridad física

Fuente: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

1. Introducción

Acceso Físico a un Dispositivo IoT

Game Over



2. Accediendo al Firmware

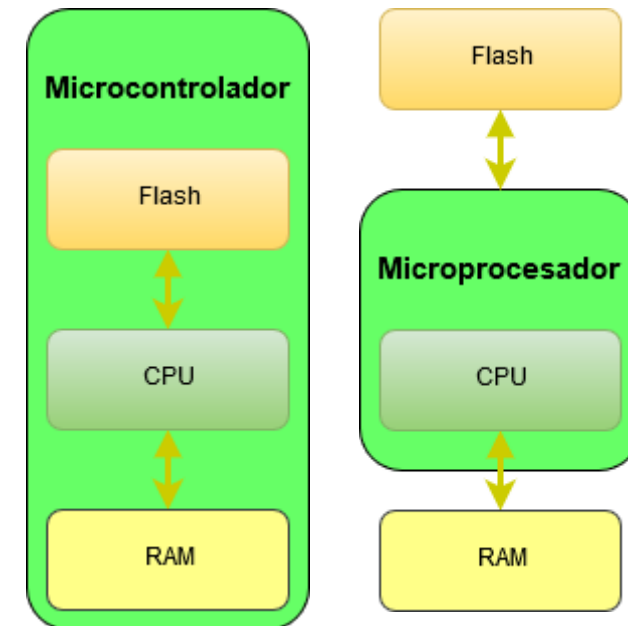
Microcontrolador vs Microprocesador

1. Microcontrolador

- Memoria RAM y *flash* en un **único chip**
- **Code Readout Protection** (CRP) protege la memoria *flash*
- Tamaño de memoria RAM y *flash* reducido
- Generalmente, no es capaz de alojar un sistema operativo

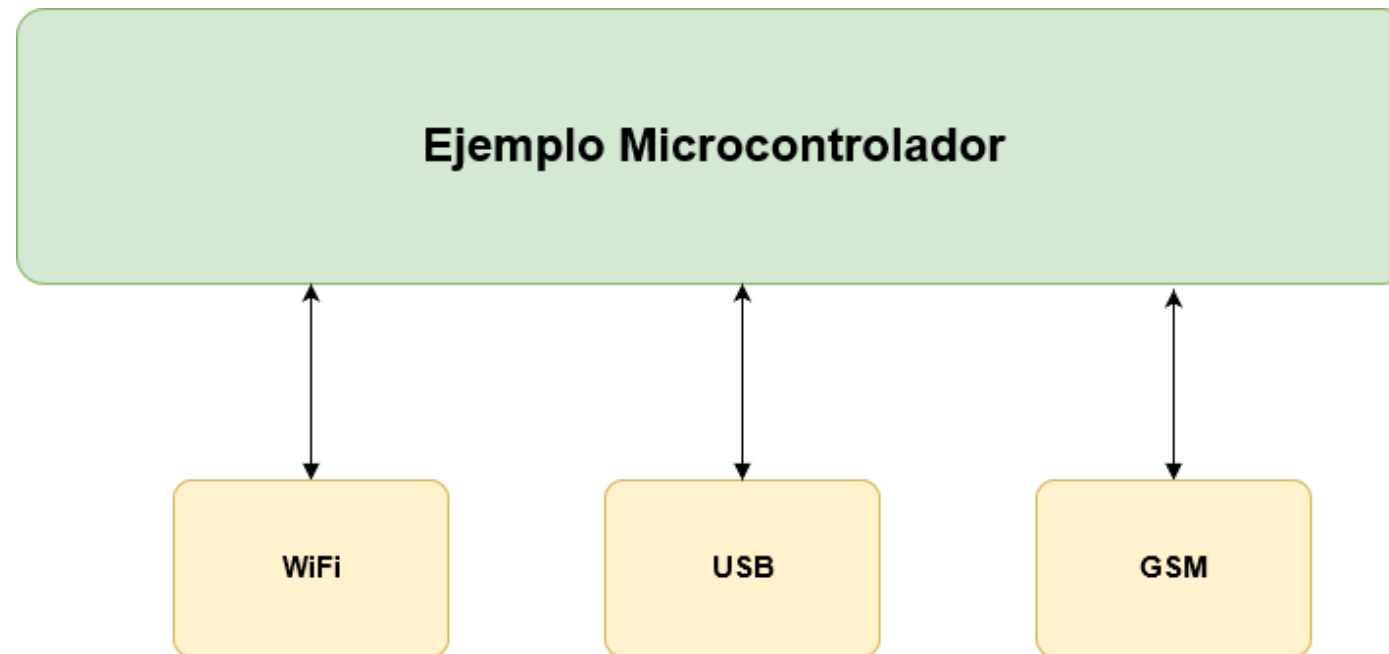
2. Microprocesador

- Memoria RAM y *flash* en **chips separados**
- Comunicaciones expuestas entre CPU y memoria *flash*
- Mayor tamaño de memoria RAM y *flash*
- Más difícil proteger los contenidos de la memoria *flash*



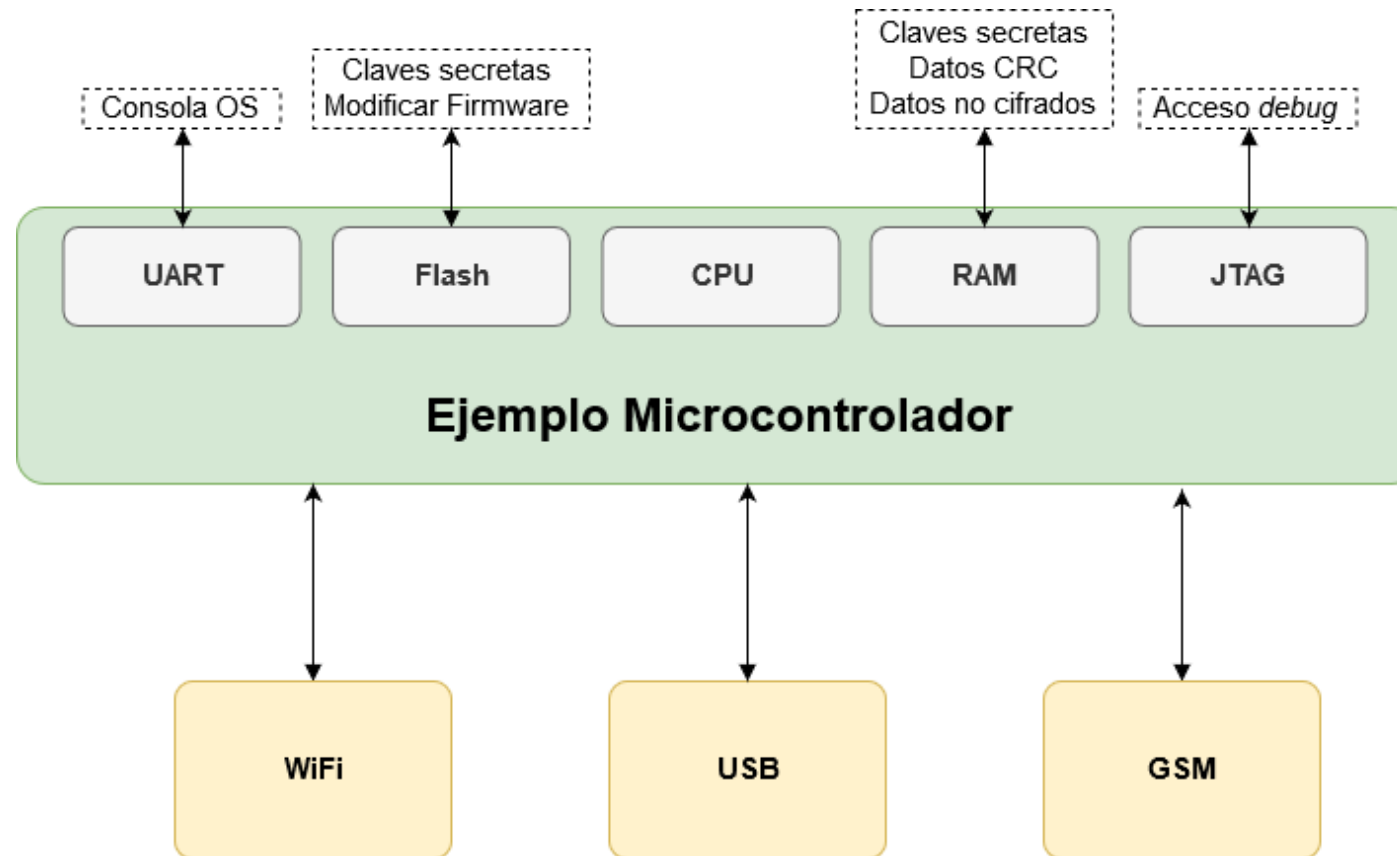
2. Accediendo al Firmware

De Caja Negra a Caja Gris



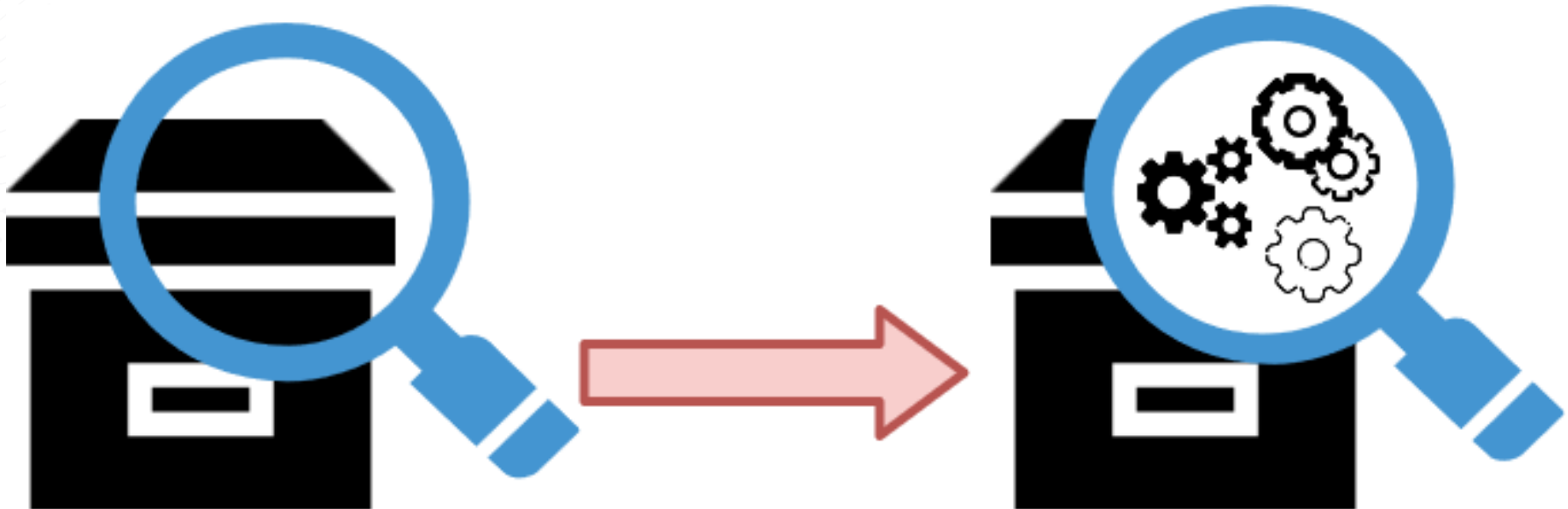
2. Accediendo al Firmware

Obtener Información o Secretos



2. Accediendo al Firmware

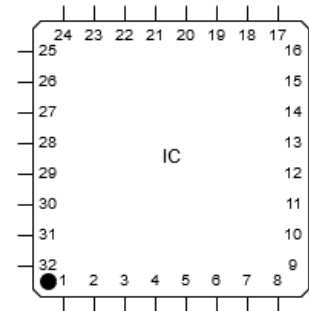
De Caja Negra a Caja Gris



2. Accediendo al Firmware

Métodos

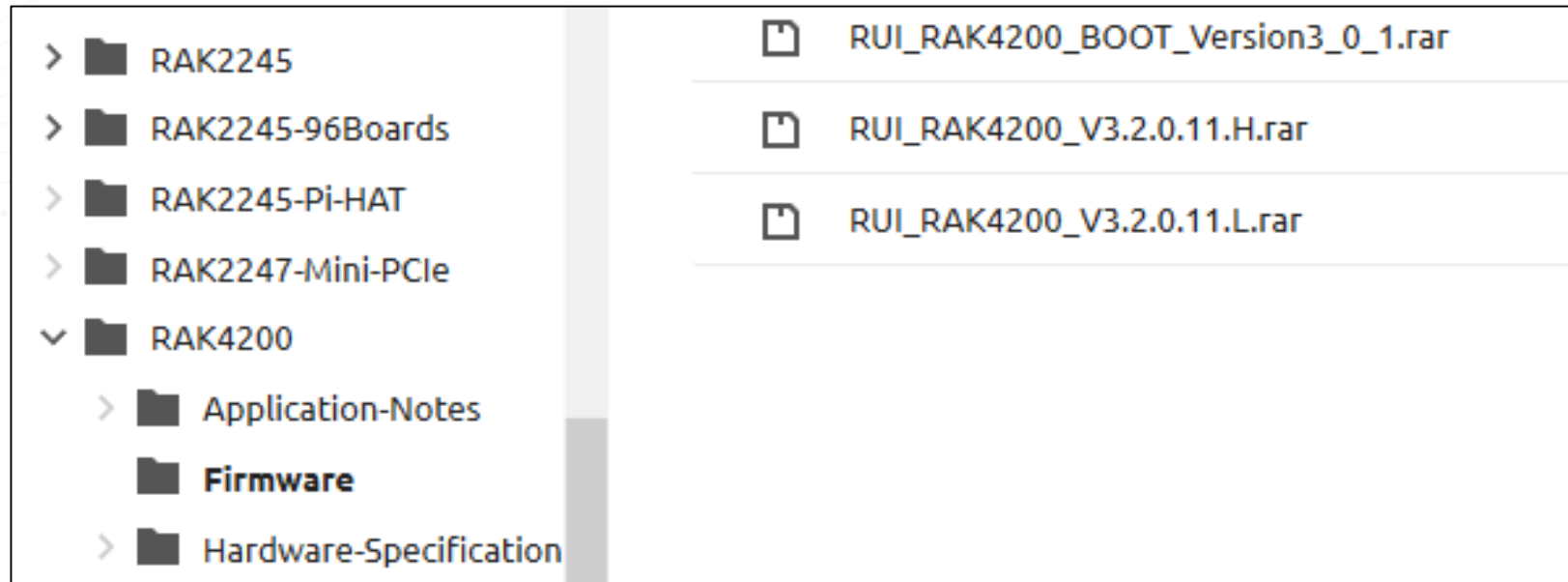
- **Internet.** Accesible al público
- Interceptar durante **actualización**
- Comunicación por **puerto de *debug***
 - SWD
 - JTAG
 - Desoldar (*chipoff attack*)



2. Accediendo al Firmware

Internet. Acceso Público

- Requiere que el *firmware* esté disponible al público

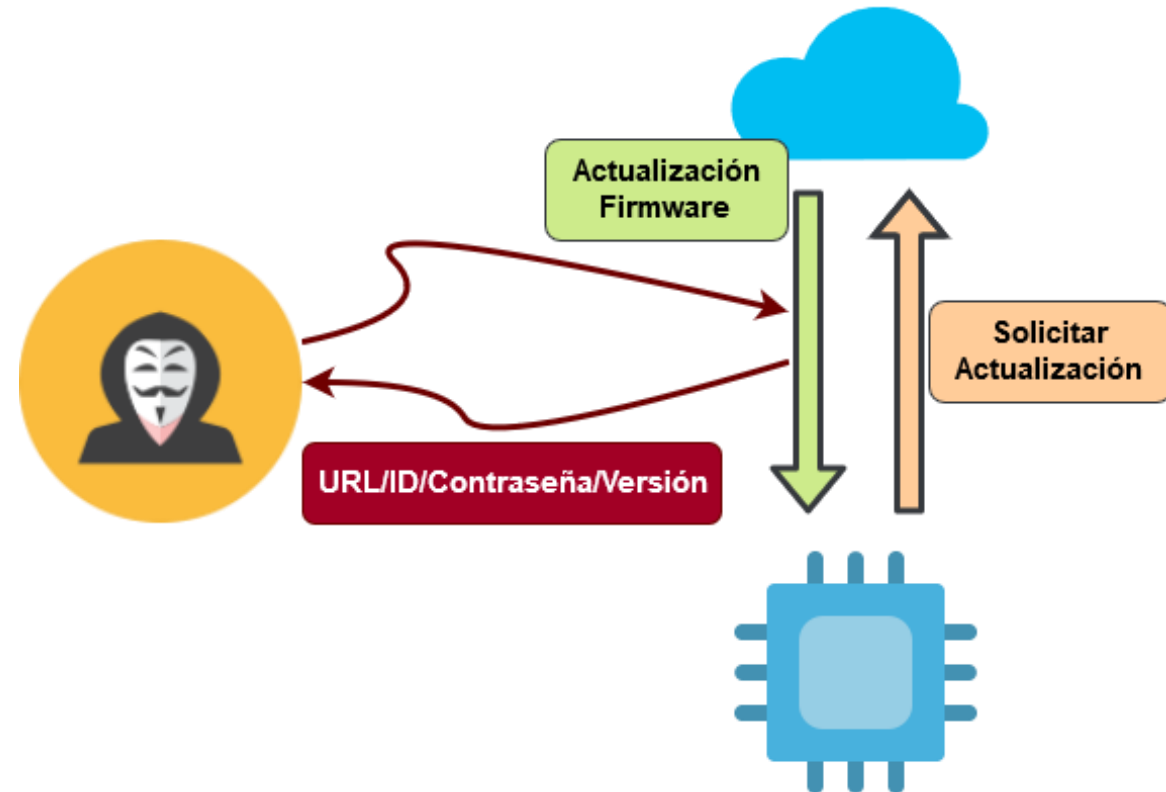


Fuente: <https://downloads.rakwireless.com/en/LoRa/RAK811/Firmware/History-Release-Verison/>

2. Accediendo al Firmware

Interceptación

- El servidor central envía una **actualización** al dispositivo
- Canal de comunicación **inseguro**
- Atacante obtiene información sensible sobre el ***firmware***



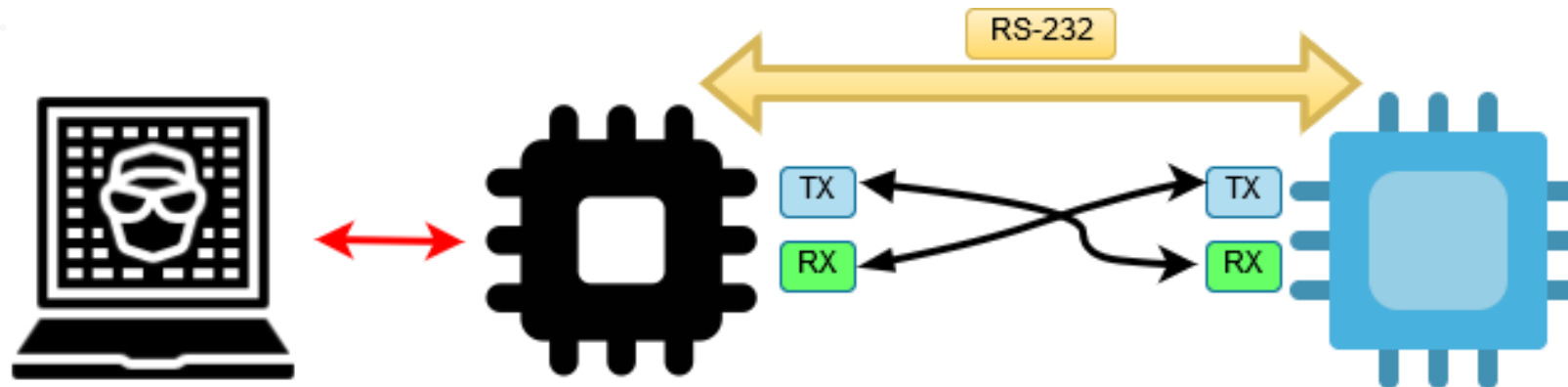
2. Accediendo al Firmware

Puerto de *Debug*

- Ataque:
 1. Identificar los pines UART
 2. Conectar dispositivo receptor
 3. Identificar la velocidad de transmisión (baudios)
 4. Utilizar la interfaz de *debugging* para obtener acceso

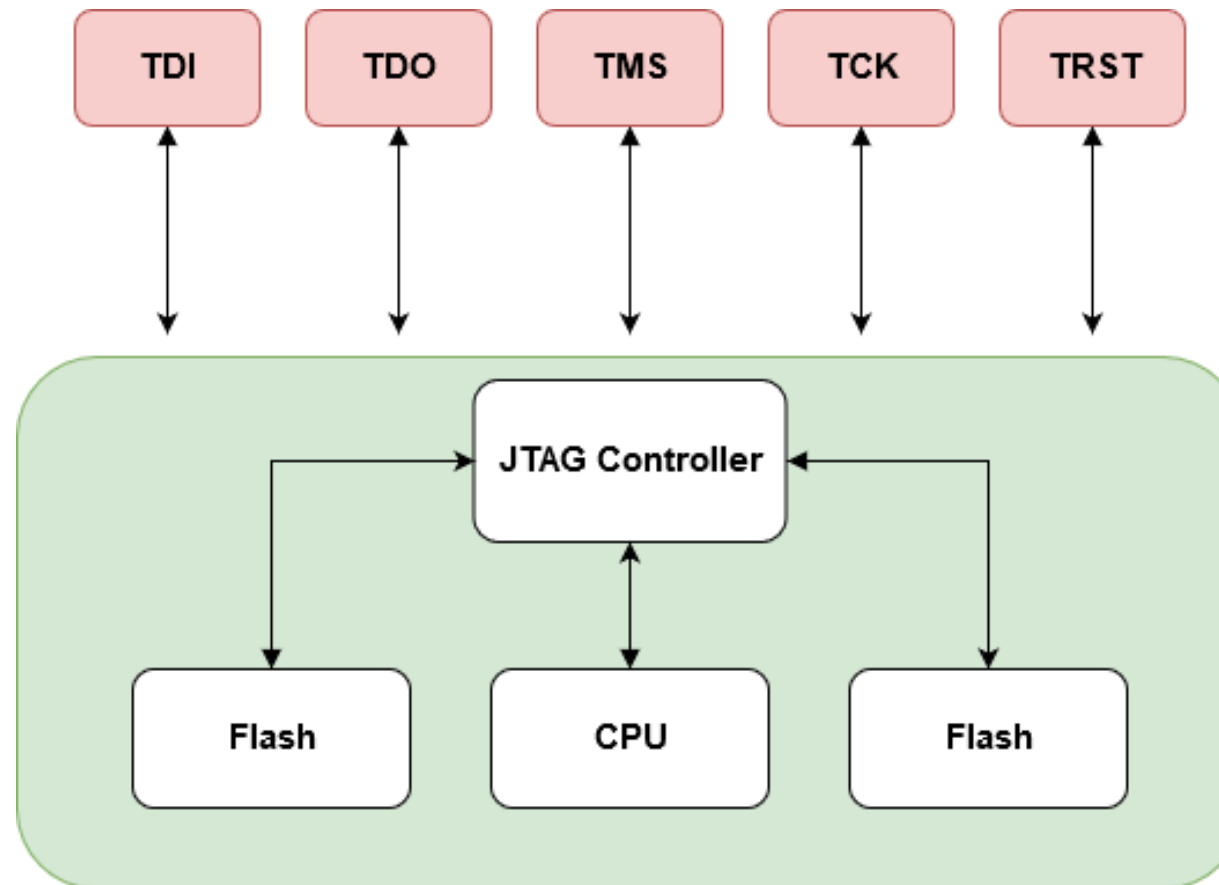


Fuente:
https://blog.securityinnovation.com/iot_uart



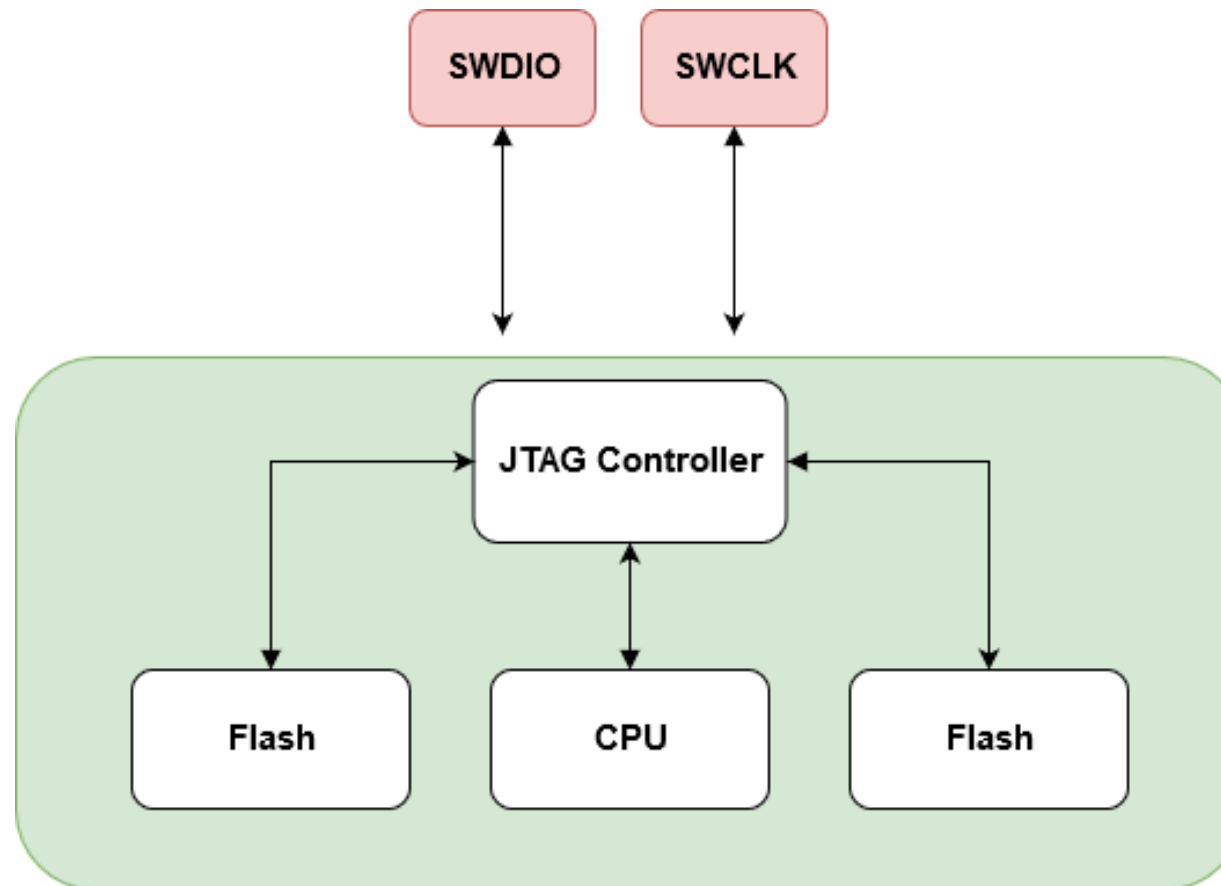
2. Accediendo al Firmware

¿Qué es JTAG?



2. Accediendo al Firmware

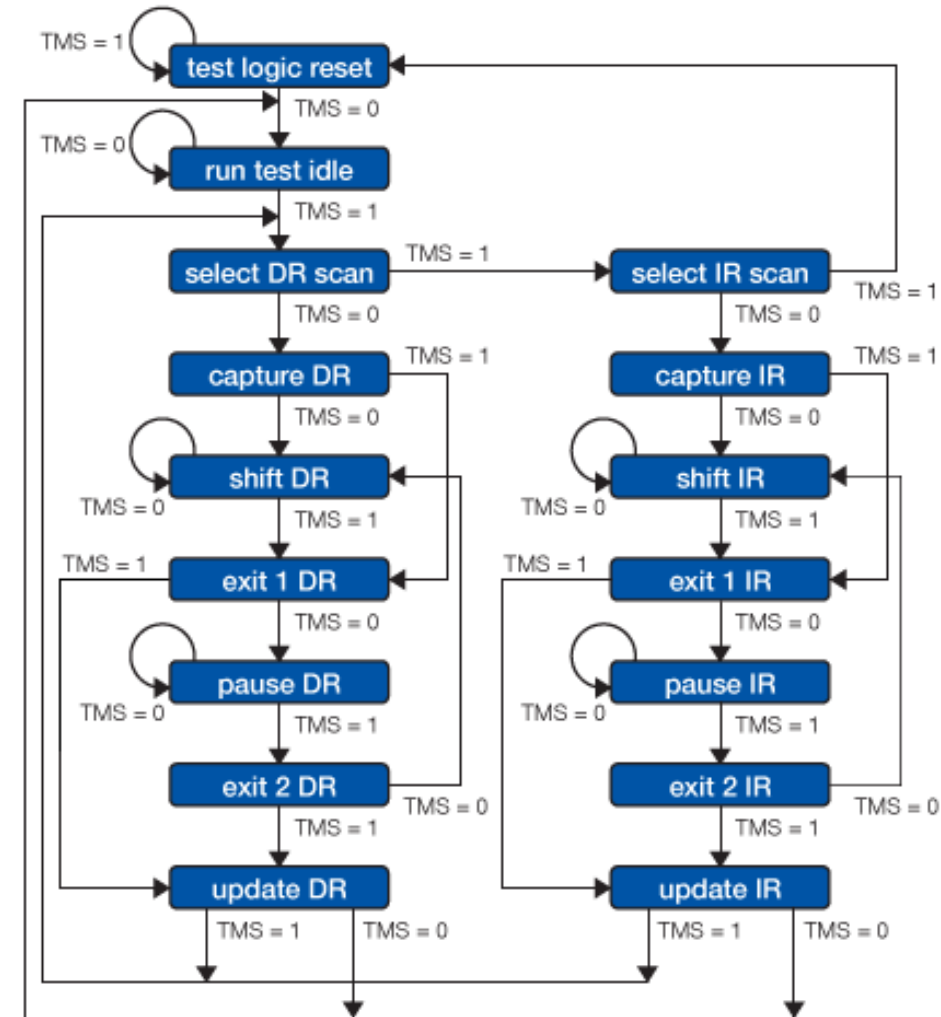
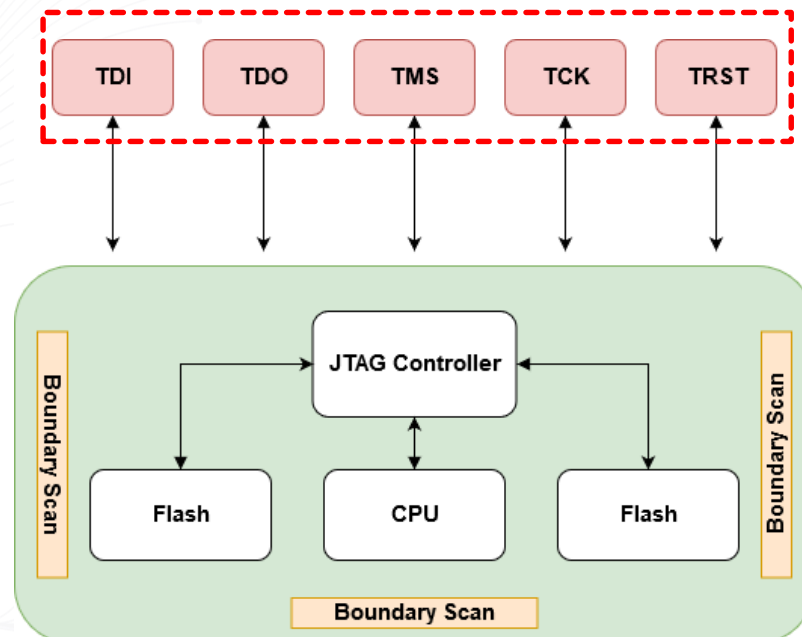
¿Qué es SWD?



2. Accediendo al Firmware

Funciones JTAG/SWD

- Siempre disponemos de:
 1. Interfaz eléctrica JTAG y SWD
 2. Máquina de estados. Usando la señal **TMS** nos movemos por el flujo.

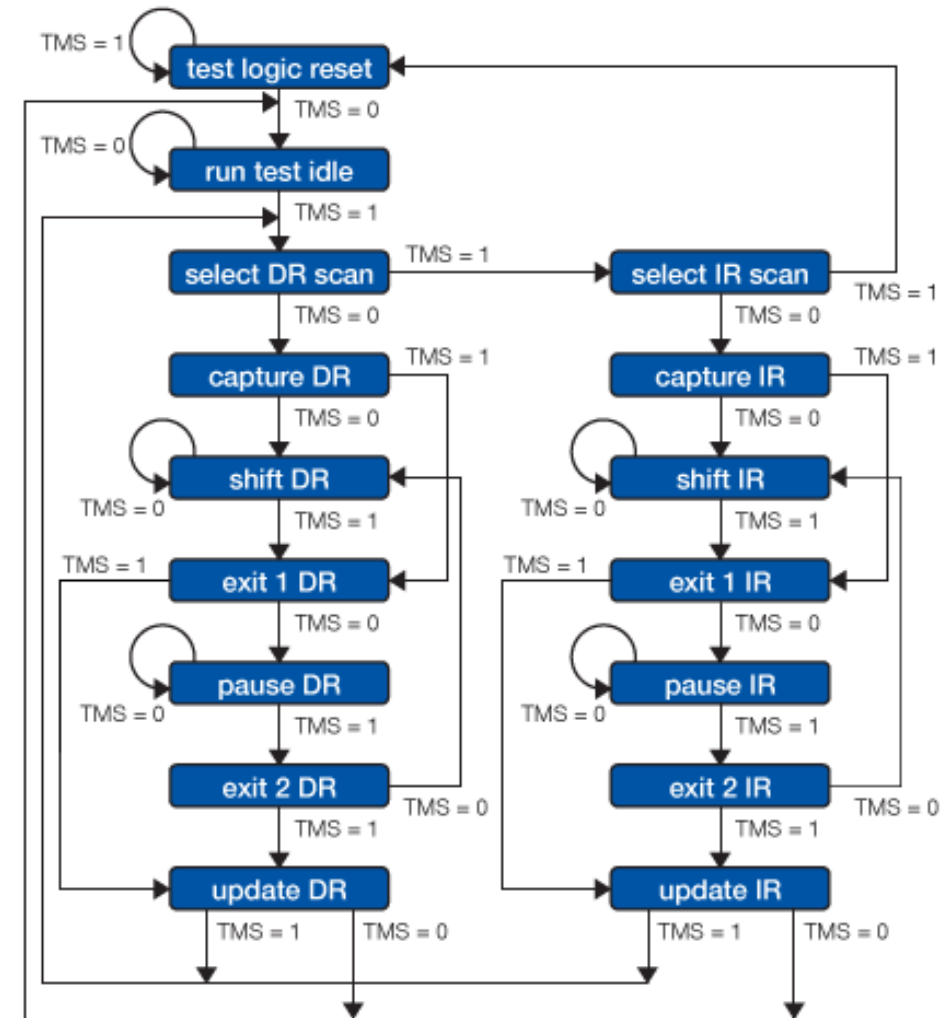
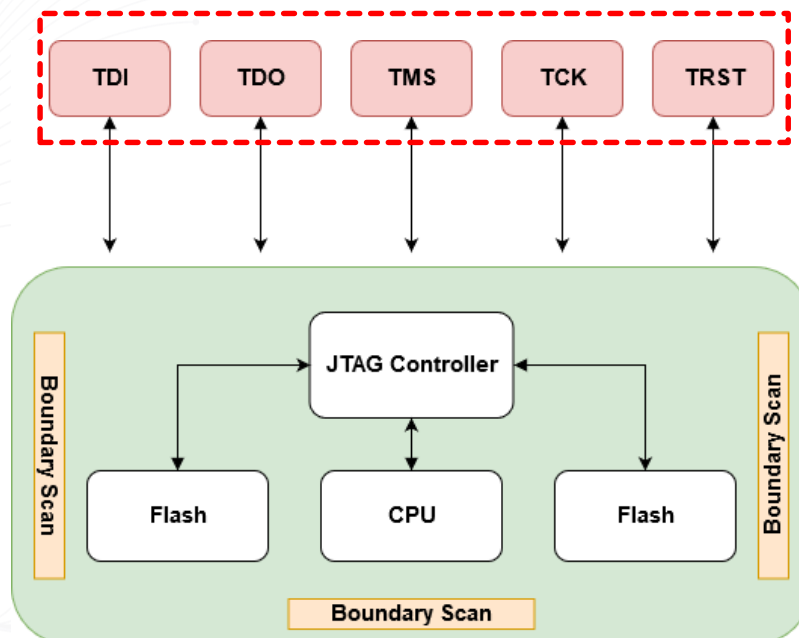


Fuente: <https://www.xjtag.com/about-jtag/jtag-a-technical-overview/>

2. Accediendo al Firmware

Funciones JTAG/SWD

- Casi siempre disponemos de:
 1. IDCODE (Identificador)
 2. *BYPASS* (TDI → TDO)
 3. *Boundary scan*. Control de pines

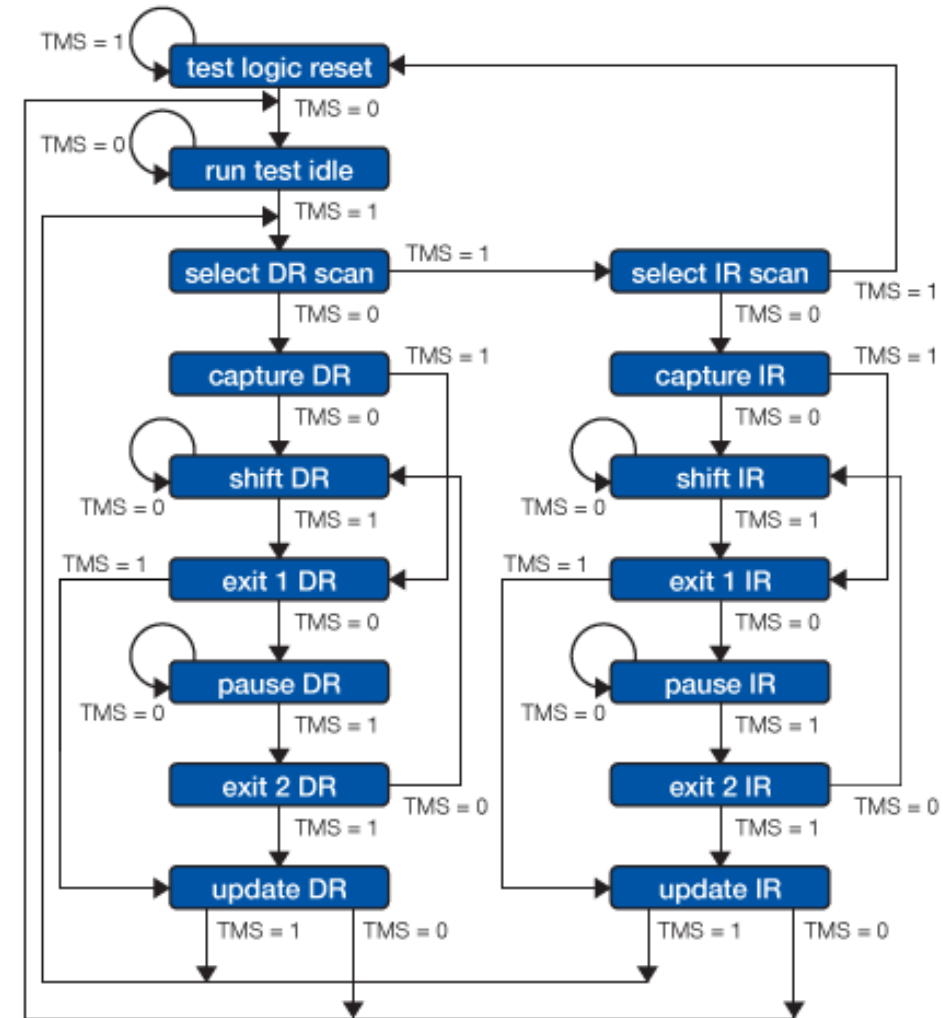
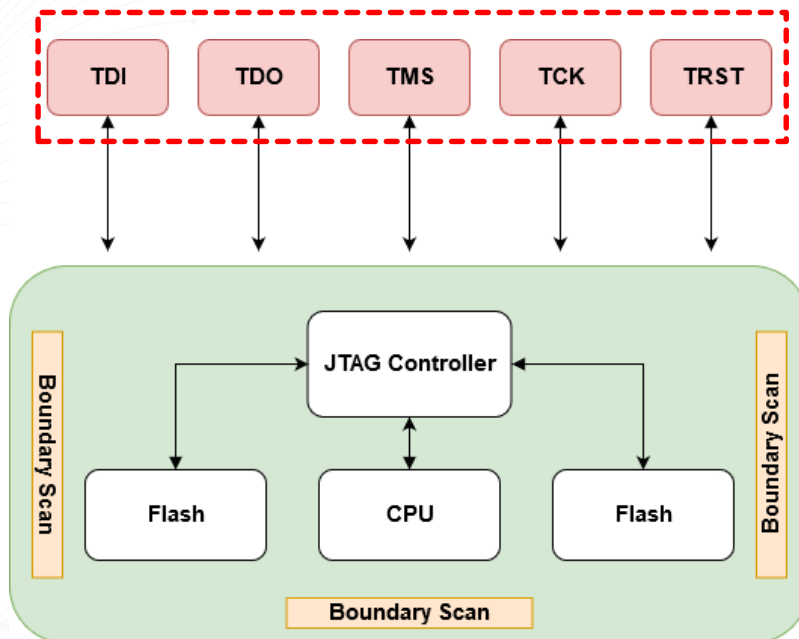


Fuente: <https://www.xjtag.com/about-jtag/jtag-a-technical-overview/>

2. Accediendo al Firmware

Funciones JTAG/SWD

- No estándar:
 - Acceso a los registros de la CPU
 - Acceso a la memoria RAM
 - Acceso a la memoria *flash*
 - Seguridad en JTAG



Fuente: <https://www.xjtag.com/about-jtag/jtag-a-technical-overview/>

3. Técnicas de Ataque a la Protección Contra Lectura

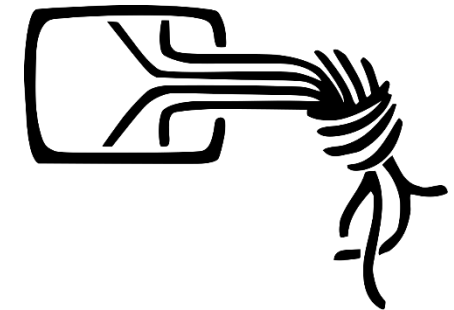
Heart of Darkness

27TH CHAOS COMMUNICATION CONGRESS IN BERLIN, DECEMBER 2010

1

Heart of Darkness - exploring the uncharted backwaters of HID iCLASS™ security

Milosch Meriac, meriac@OpenPCD.de



Abstract—This paper provides detailed information on iCLASS™ reader and key security. It explains the security problems found without revealing the extracted secret keys (DES authentication Key and the 3DES data encryption key for iCLASS™ Standard Security cards).

The chosen approach of not releasing the encryption and authentication keys gives iCLASS vendors and customers an important headstart to update readers and cards to High Security mode in order to stop attackers from forging, reading and cloning iCLASS Standard Security cards.

This paper also explains, how Standard Security and High Security keys were extracted from a RW400 reader without leaving visible traces.

Detailed suggestions to improve system security can be found in chapter IX.

The protocol security aspects of the iCLASS RFID protocol will be presented separately at the public 27C3 talk and thus will not be duplicated here in this paper.

II. iCLASS SECURITY

*Do you want the convenience
of receiving preprogrammed cards
that are ready for use? No problem -
trust HID to manage your keys! -*

Fuente: <https://www.openpcd.org/dl/HID-iCLASS-security.pdf>

Heart of Darkness

- PIC18FXX2/XX8
- Dividido en bloques
- Configuración **individual** para cada bloque

[illegible]

Fuente: <http://ww1.microchip.com/downloads/en/DeviceDoc/39576b.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness

3.1 High Voltage ICSP Bulk Erase

Erasing code or data EEPROM is accomplished by writing an “erase option” to address 3C0004h. Code memory may be erased portions at a time, or the user may erase the entire device in one action. “Bulk Erase” operations will also clear any code protect settings associated with the memory block erased. Erase options are detailed in Table 3-1.

TABLE 3-1: BULK ERASE OPTIONS

Description	Data
Chip Erase	80h
Erase Data EEPROM	81h
Erase Boot Block	83h
Erase Panel 1	88h
Erase Panel 2	89h
Erase Panel 3	8Ah
Erase Panel 4	8Bh

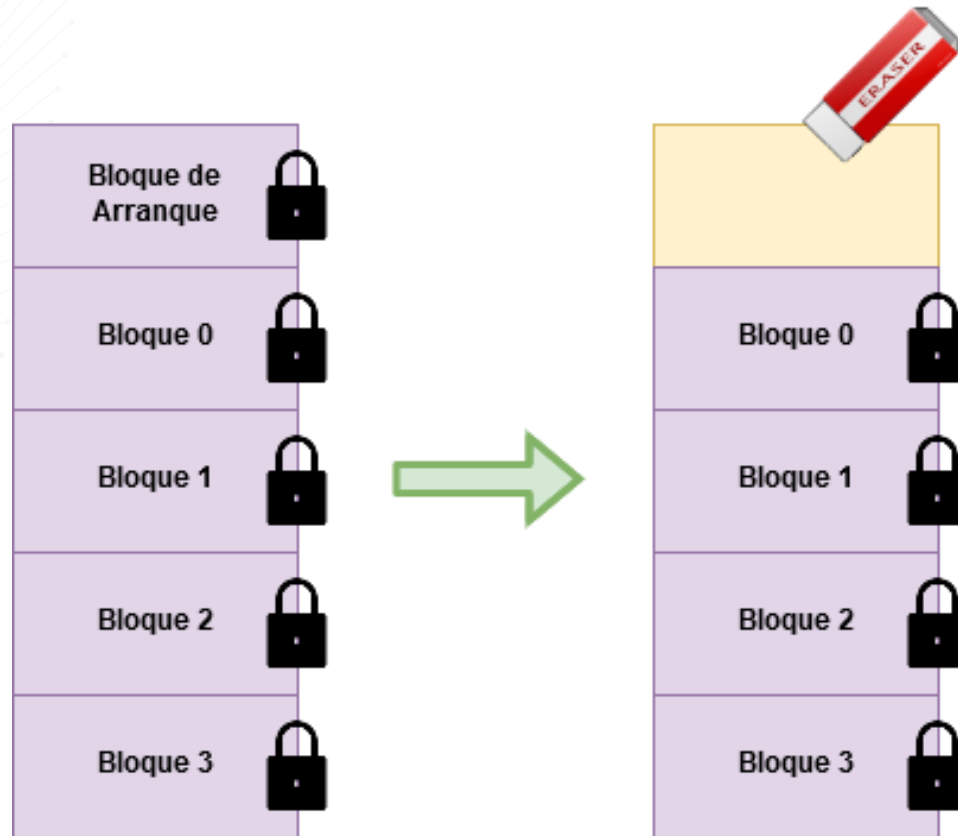
3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



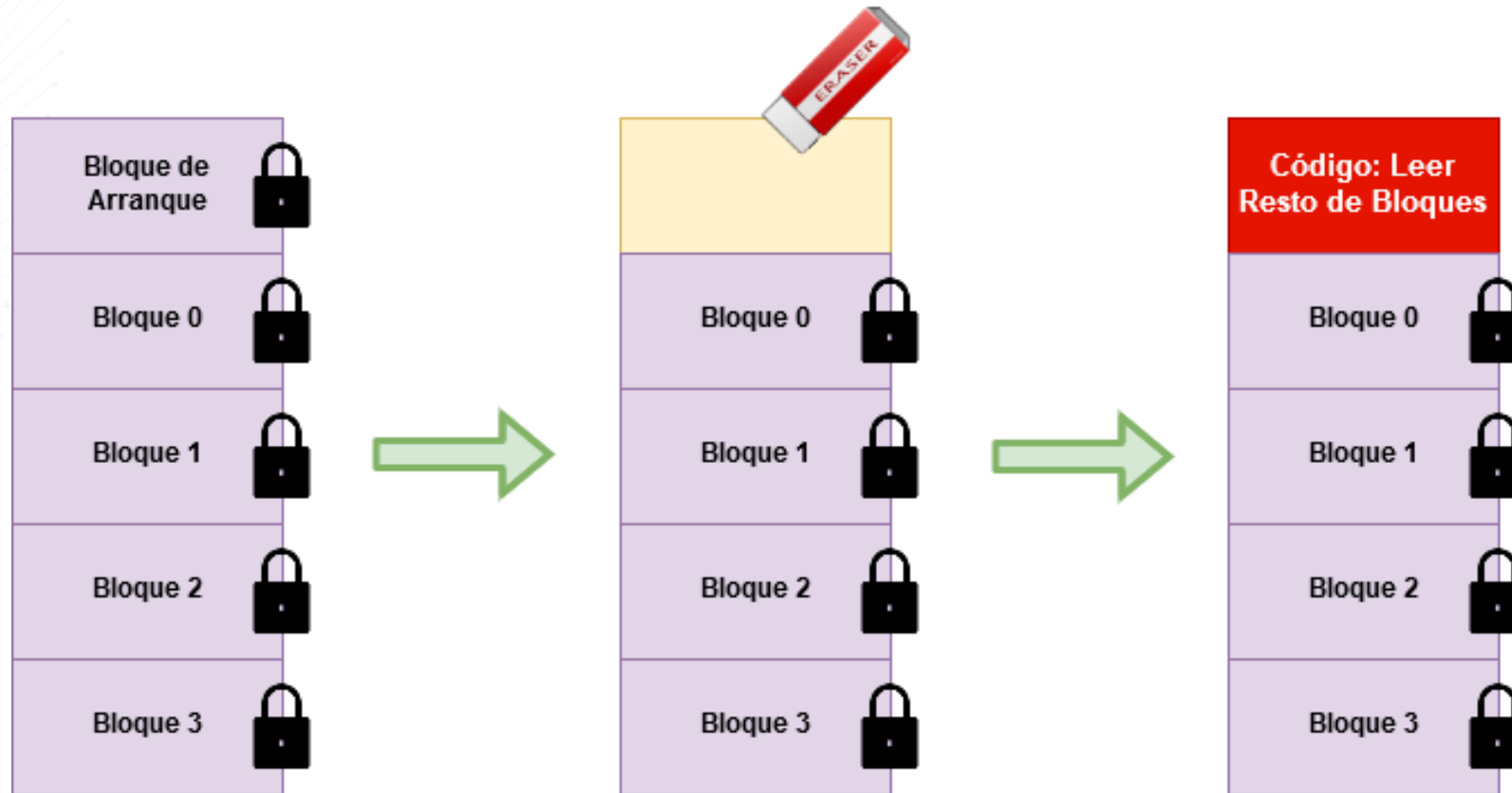
3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



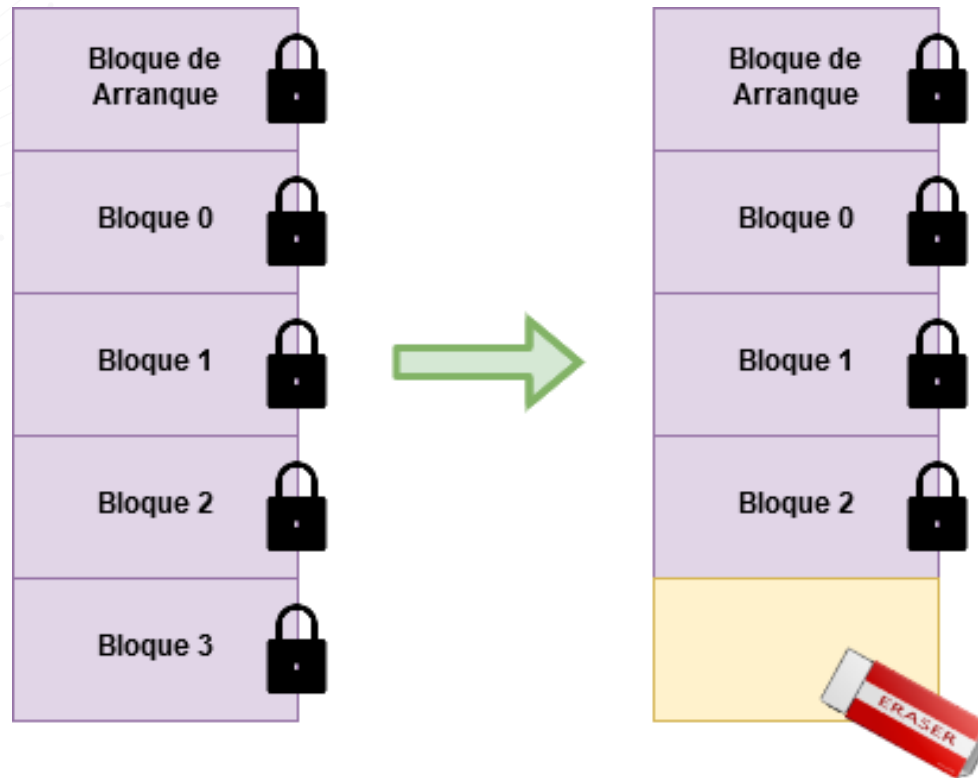
3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



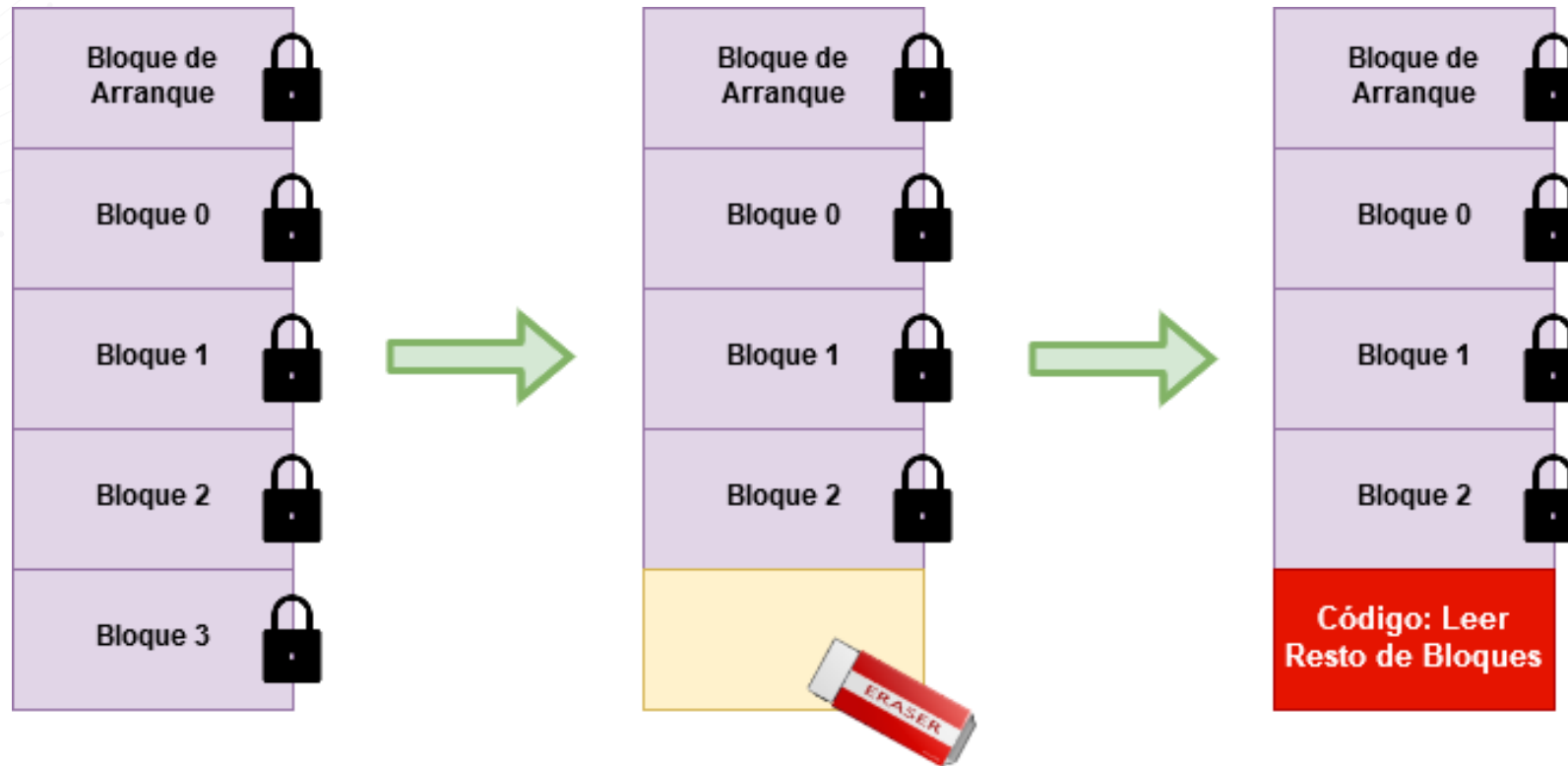
3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



3. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



EM359x

Ember[®] EM359x Reference Manual

In general, if read protection is active then write protection should also be active. This prevents an attacker from reprogramming flash with malicious code that could readout the flash after the debugger is disconnected. To obtain fully protected flash, both read protection and write protection should be active.

Fuente: <https://www.silabs.com/documents/public/reference-manuals/EM359x-RM.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

UV-C *Security Fuse Erase*



Shedding too much Light on a Microcontroller's Firmware Protection

Johannes Obermaier

Fraunhofer Institute AISEC

johannes.obermaier@aisec.fraunhofer.de

Stefan Tatschner

Fraunhofer Institute AISEC

stefan.tatschner@aisec.fraunhofer.de

Fuente: <https://www.usenix.org/system/files/conference/woot17/woot17-paper-obermaier.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

UV-C *Security Fuse Erase*

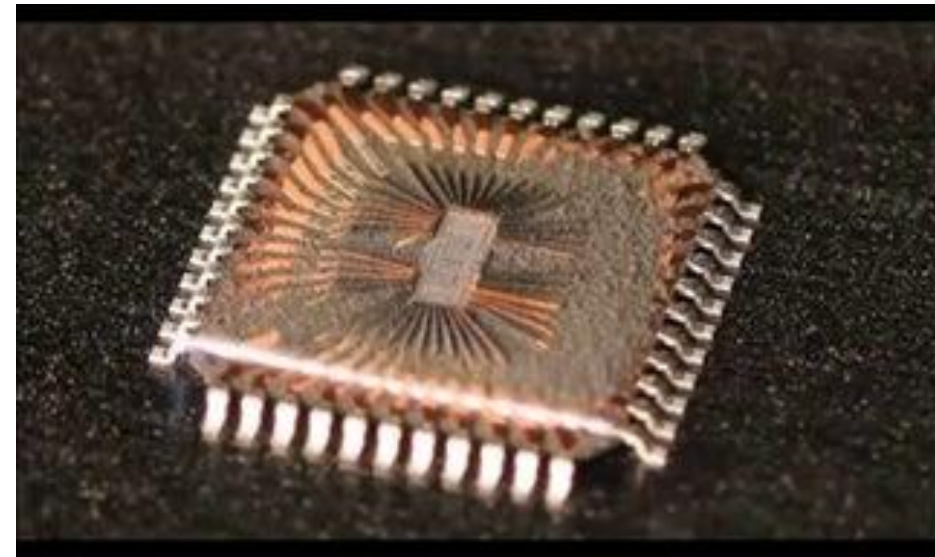
Table 6. Flash memory read protection status

RDP byte value	RDP complement value	Read protection level
0xAA	0x55	Level 0 (ST production configuration)
Any value except 0xAA or 0xCC	Any value (not necessarily complementary) except 0x55 and 0x33	Level 1
0xCC	0x33	Level 2

Fuente: <https://usermanual.wiki/Pdf/STM32Fxreference20manual.1209178456/view>

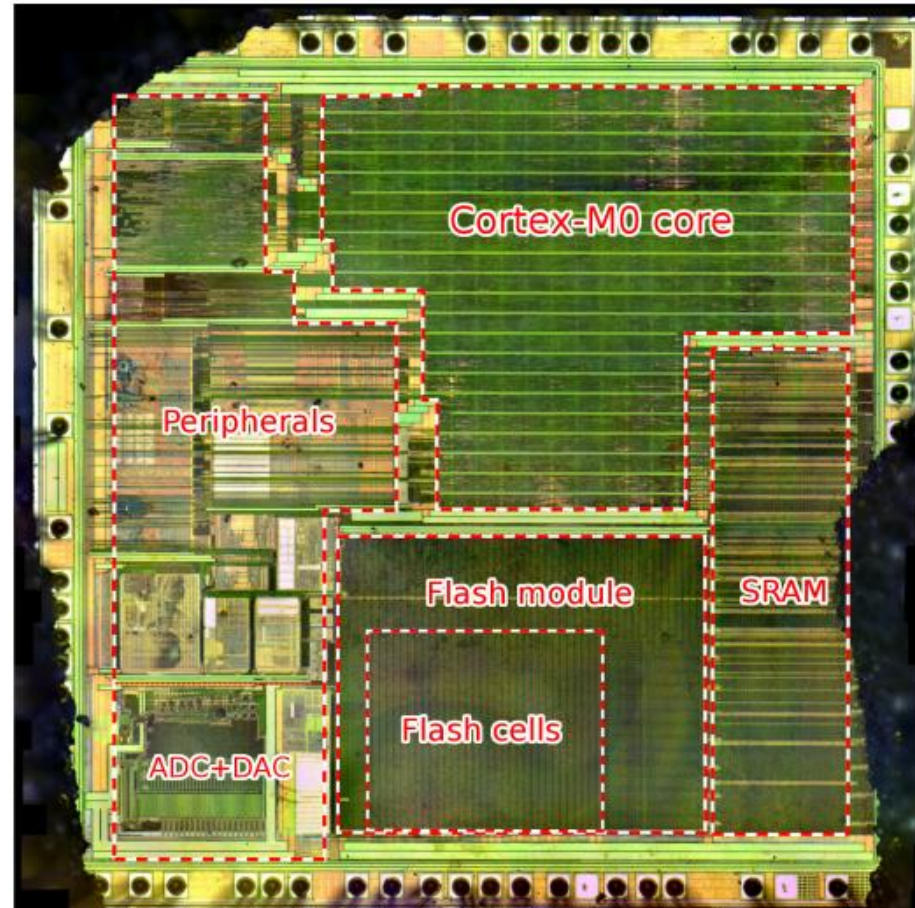
3. Técnicas de Ataque a la Protección Contra Lectura

UV-C Security Fuse Erase



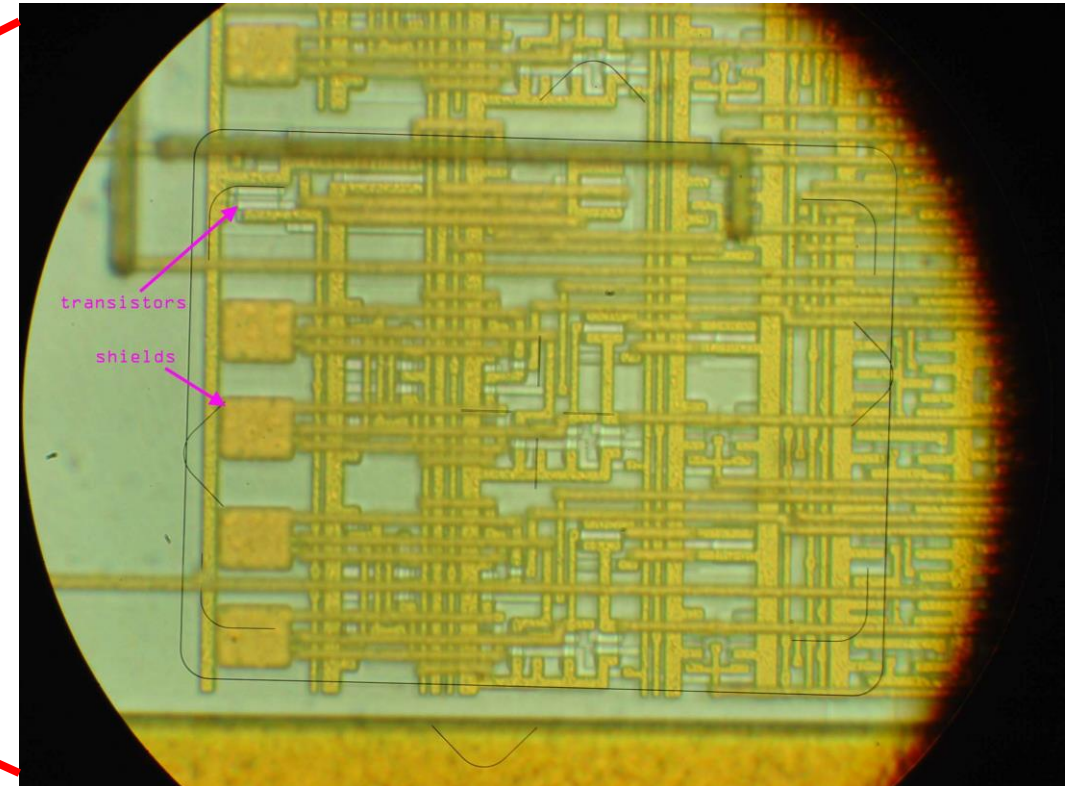
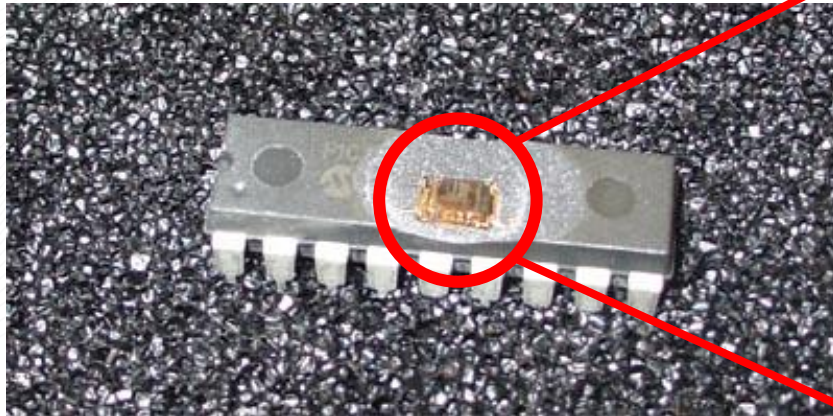
3. Técnicas de Ataque a la Protección Contra Lectura

UV-C *Security Fuse Erase*



3. Técnicas de Ataque a la Protección Contra Lectura

UV-C Security Fuse Erase



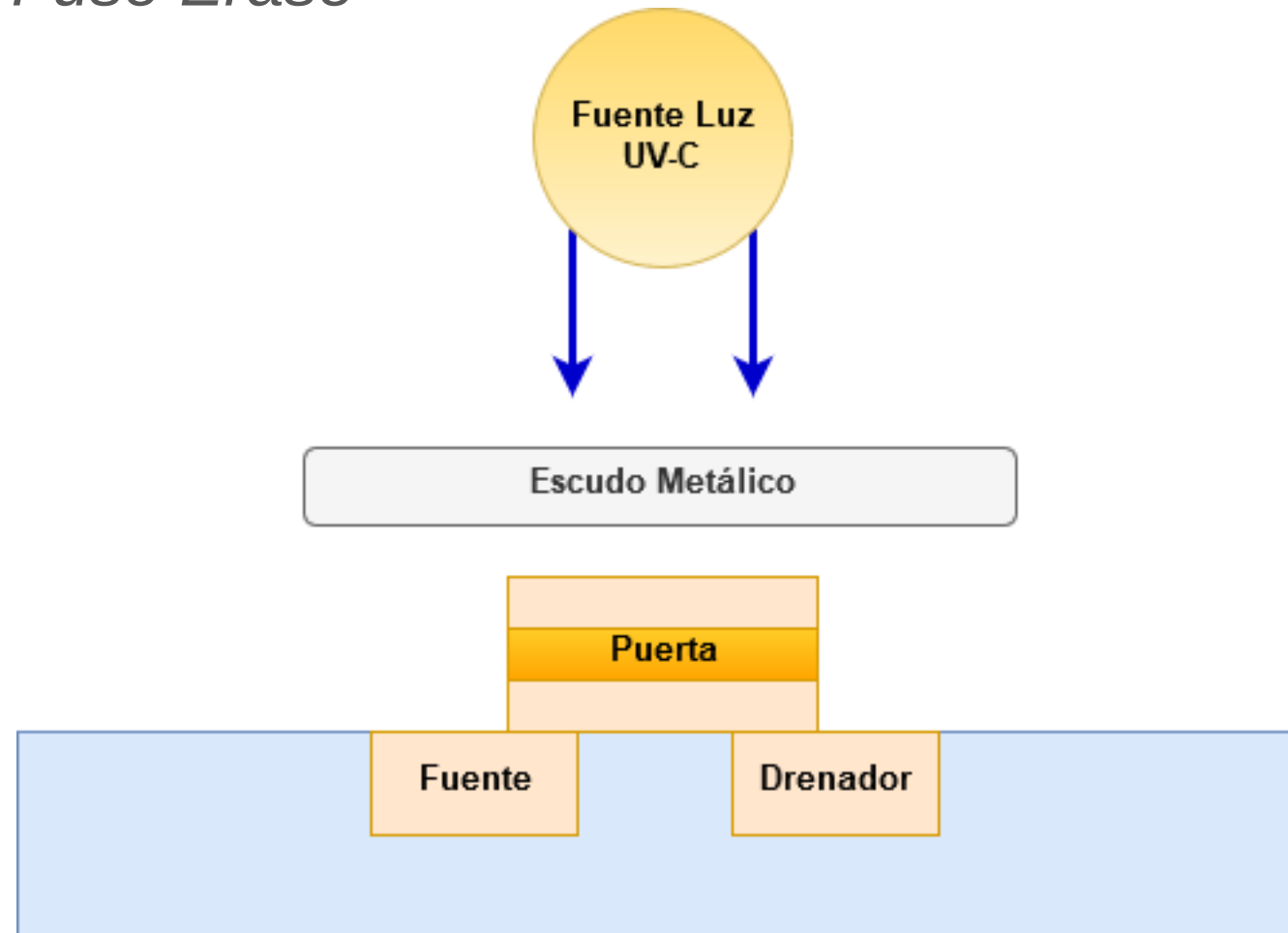
3. Técnicas de Ataque a la Protección Contra Lectura

UV-C Security Fuse Erase



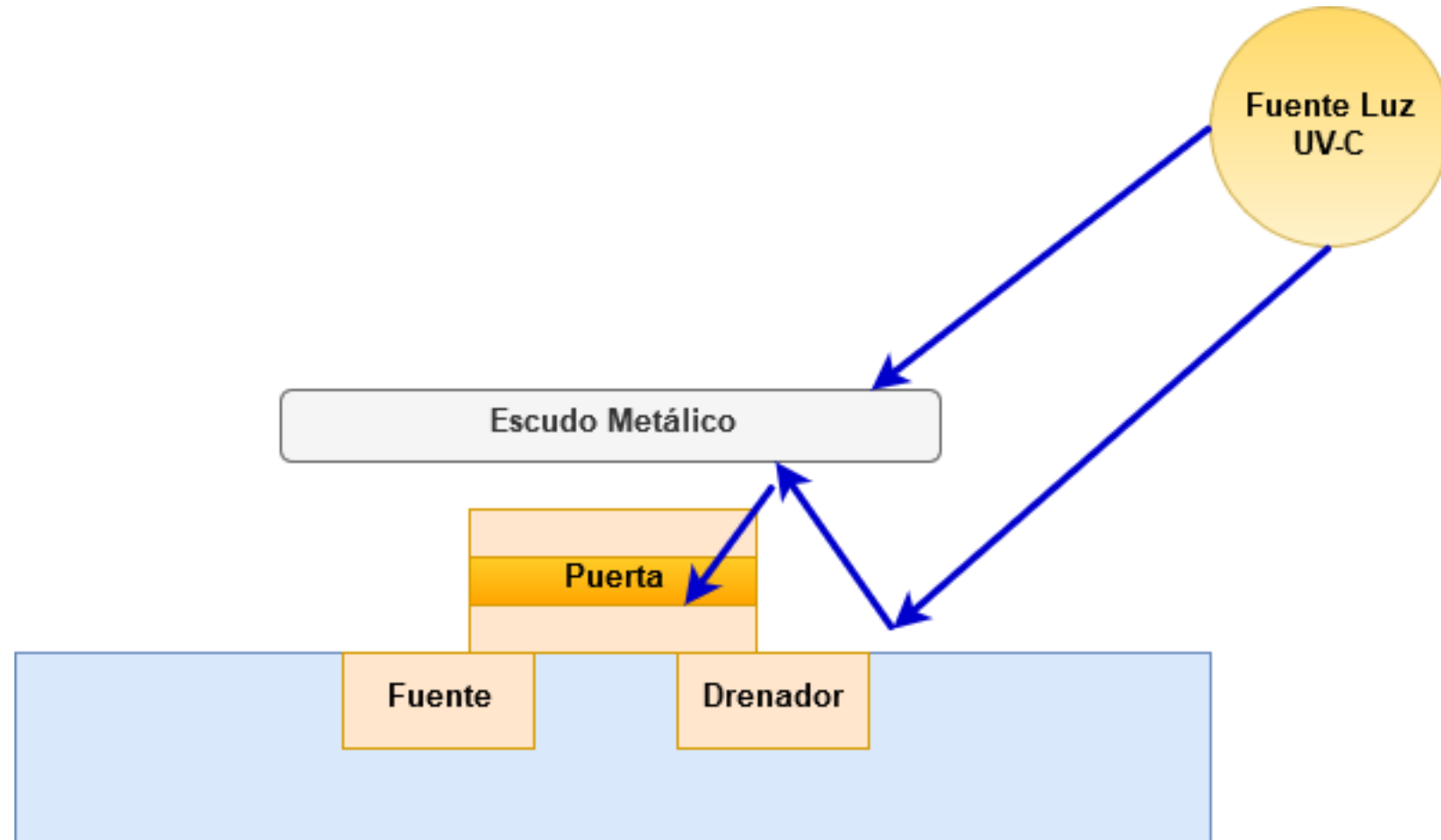
3. Técnicas de Ataque a la Protección Contra Lectura

UV-C Security Fuse Erase



3. Técnicas de Ataque a la Protección Contra Lectura

UV-C Security Fuse Erase



3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



Shedding too much Light on a Microcontroller's Firmware Protection

Johannes Obermaier

Fraunhofer Institute AISEC

johannes.obermaier@aisec.fraunhofer.de

Stefan Tatschner

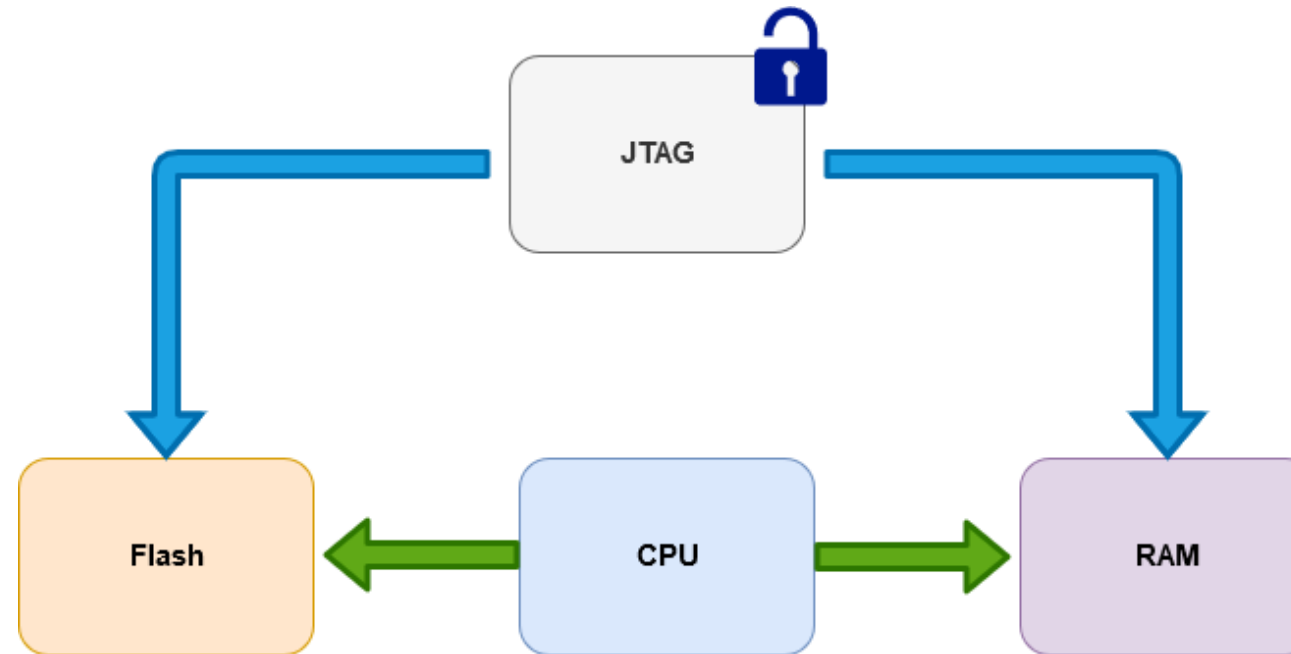
Fraunhofer Institute AISEC

stefan.tatschner@aisec.fraunhofer.de

Fuente: <https://www.usenix.org/system/files/conference/woot17/woot17-paper-obermaier.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

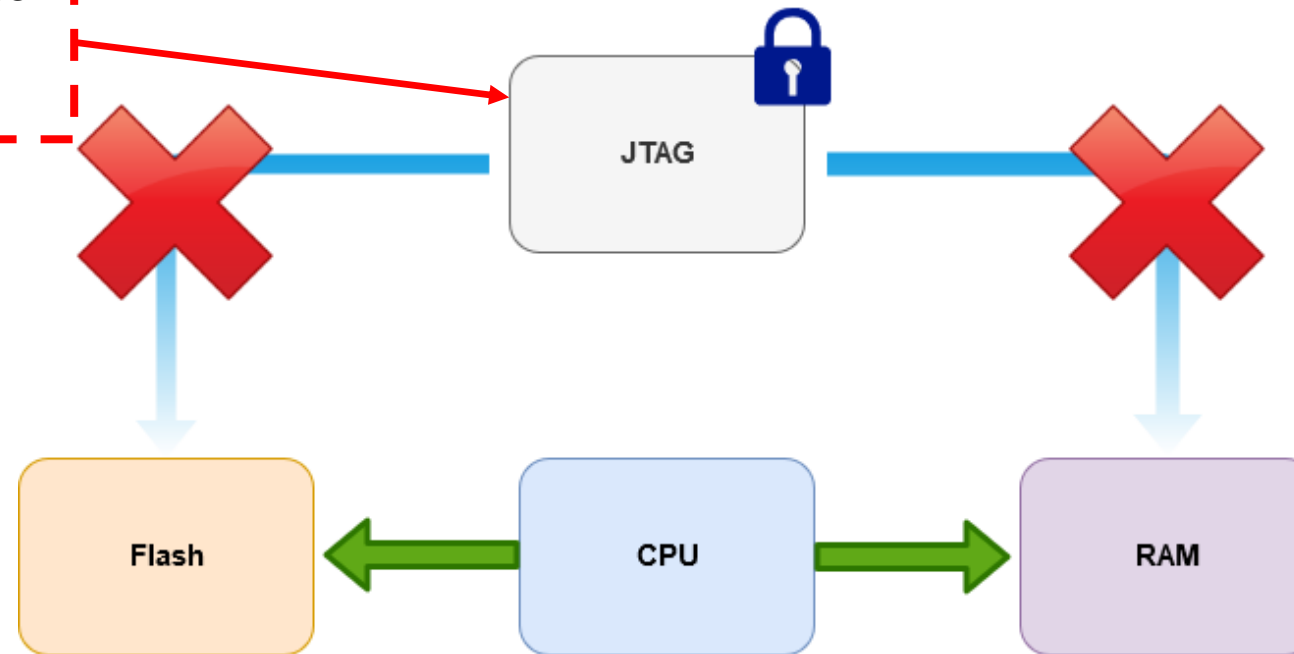
Flash Lock Race



3. Técnicas de Ataque a la Protección Contra Lectura

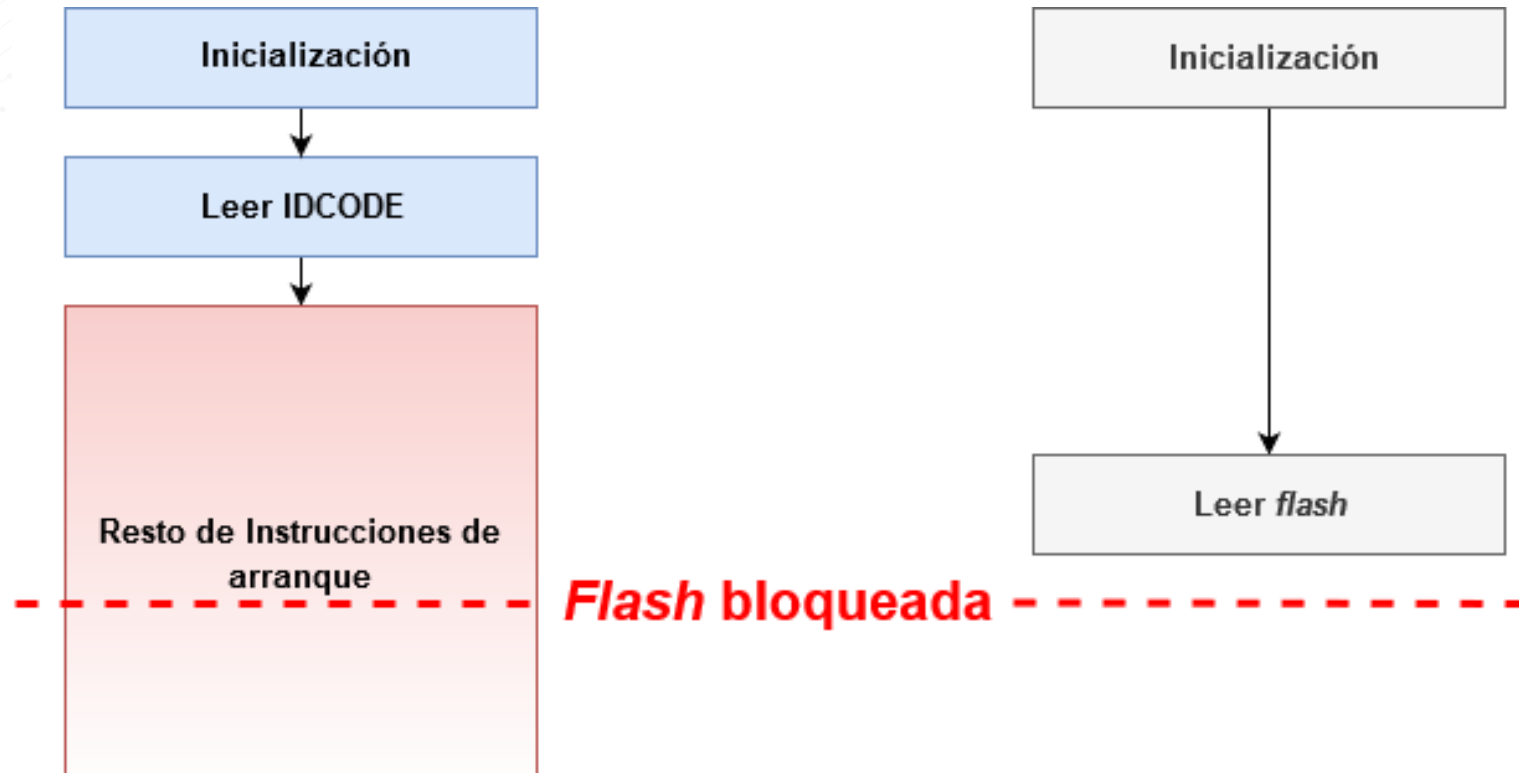
Flash Lock Race

No conocemos los
detalles de la
implementación



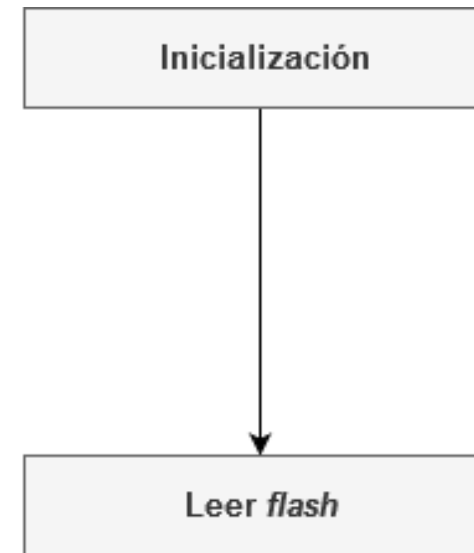
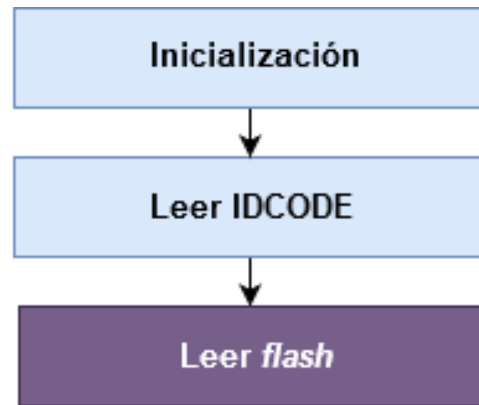
3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



----- **Flash bloqueada** -----

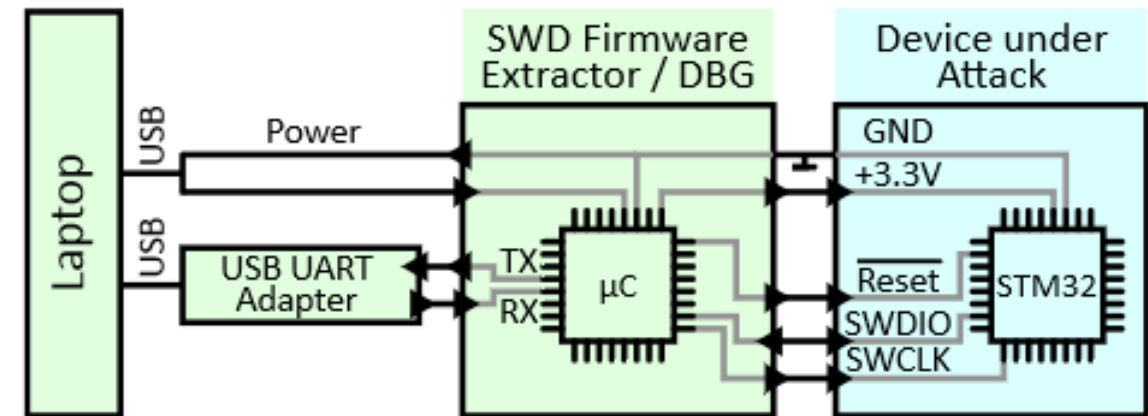
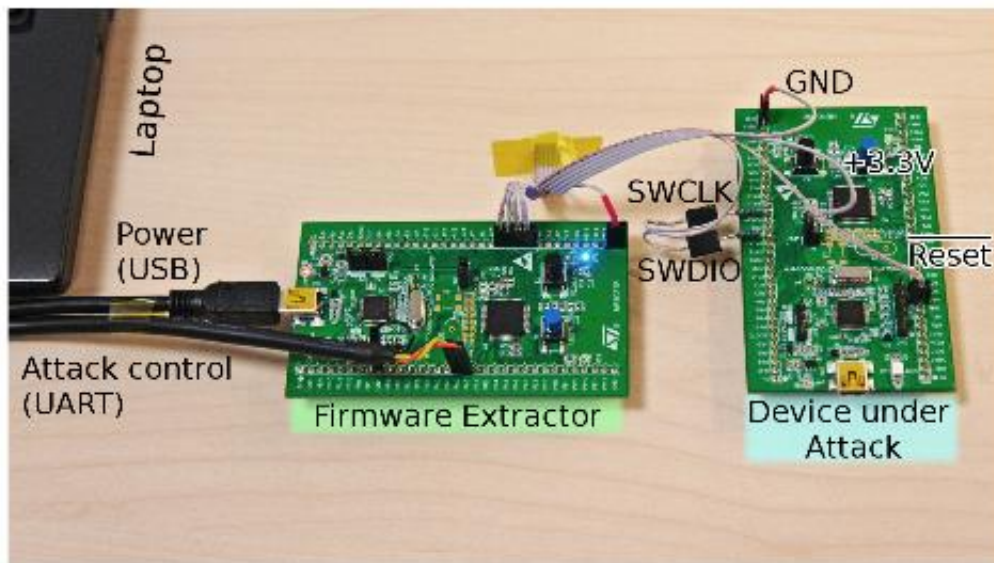
3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race

- Minimización de SWD
- Solicitar acceso a la memoria *flash*
- Condición de carrera, debemos leer la memoria antes de que esta sea bloqueada
 1. Reiniciamos el sistema
 2. Inicializamos la interfaz de *debug*
 3. Establecemos la dirección de memoria que queremos leer
 4. Leemos la memoria
 5. Si tenemos éxito, repetimos el proceso para subsecuentes secciones de memoria

3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



Fuente: <https://www.usenix.org/system/files/conference/woot17/woot17-paper-obermaier.pdf>

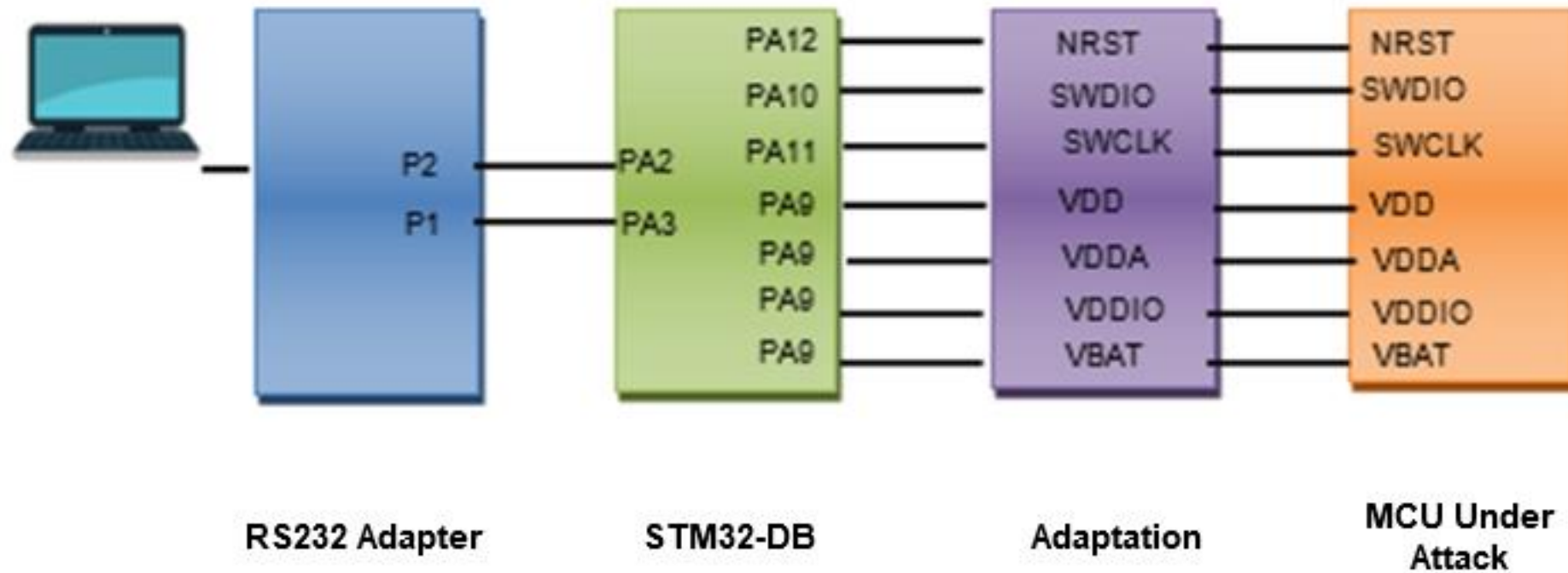
3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



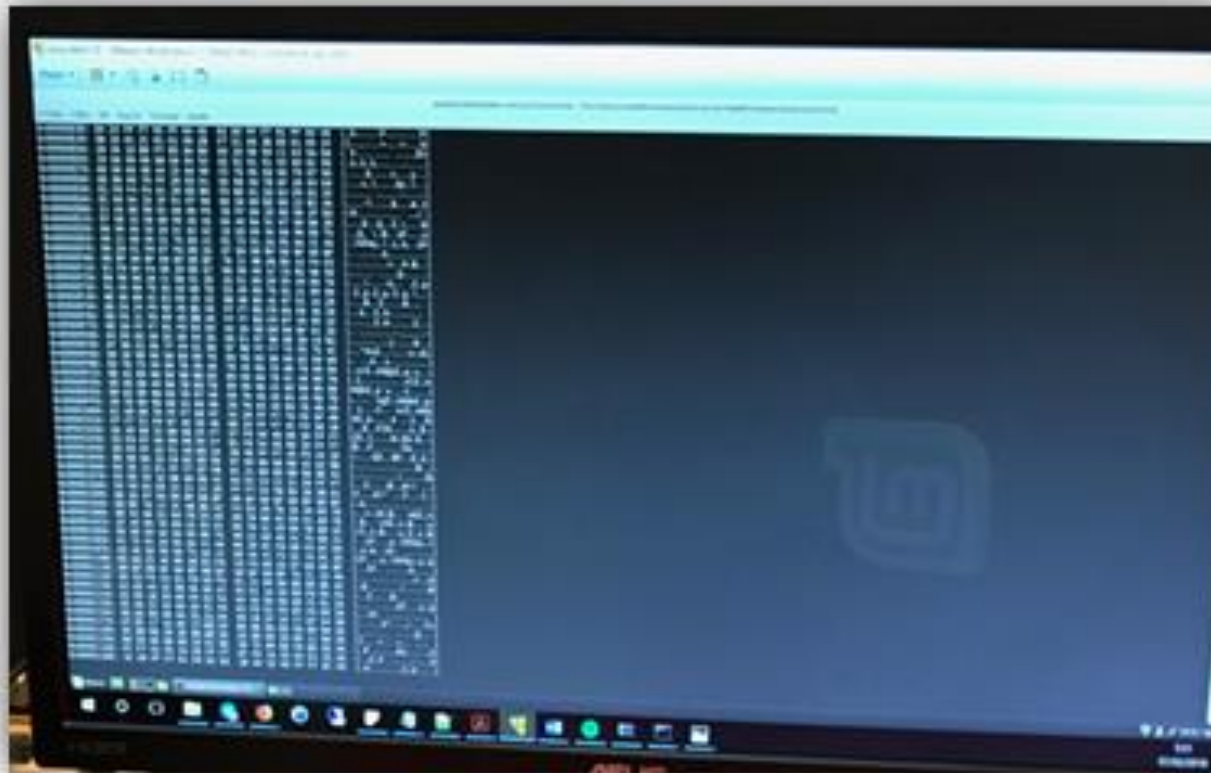
3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



3. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



3. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping



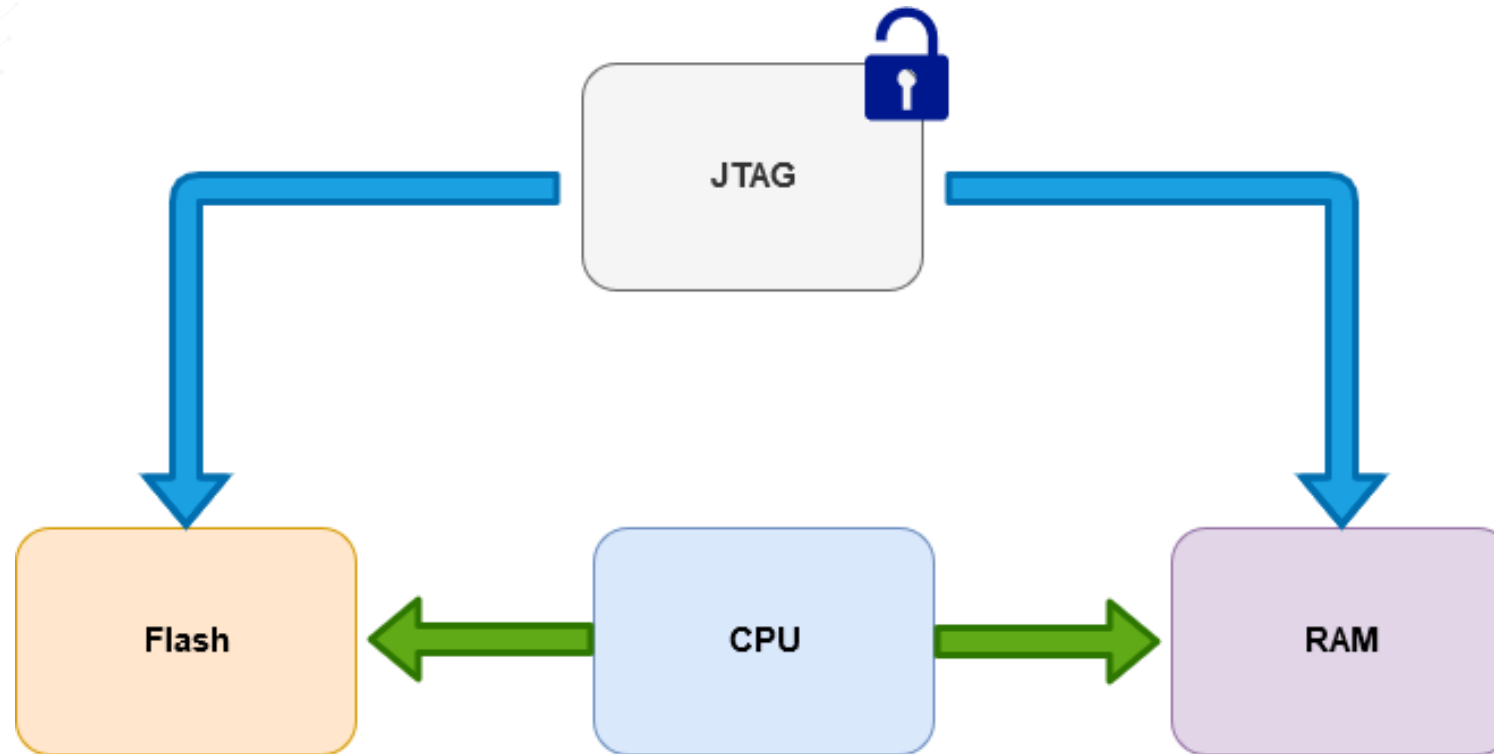
Idols with Feet of Clay: On the Security of Bootloaders and Firmware Updaters for the IoT

Lionel Morel and Damien Couroussé
Univ. Grenoble Alpes, CEA, List
F-38000 Grenoble, France
Email: {lionel.morel, damien.courousse}@cea.fr

Fuente: <https://damien.courousse.fr/pdf/2019-NEWCAS-preprint.pdf>

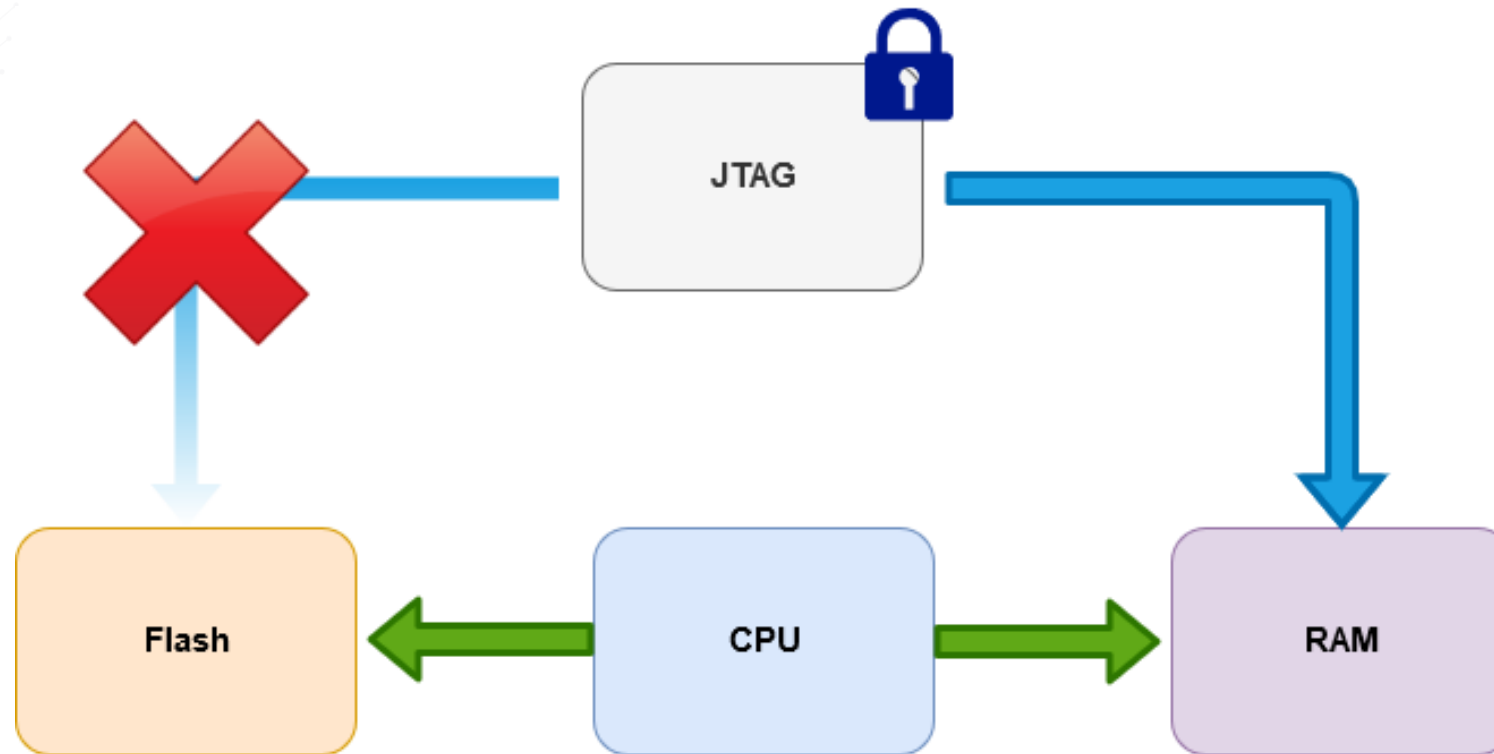
3. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping



3. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping

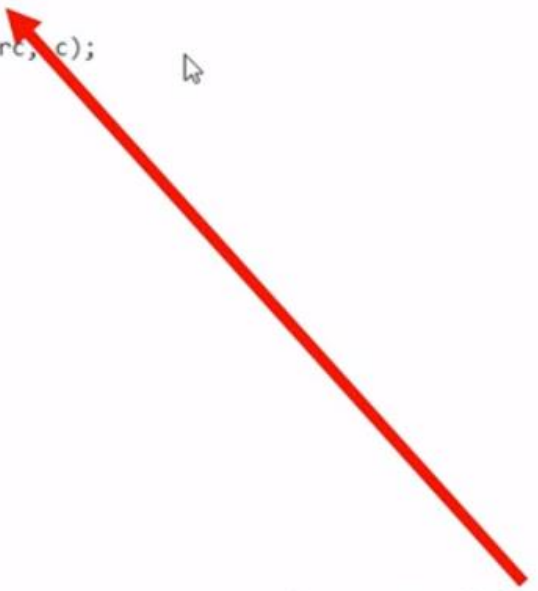


3. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping

- *Bootloader* comprueba la integridad de la memoria *flash* usando **CRC**
- Dependiendo de la implementación, el resultado del CRC es **almacenado en SRAM**
- CRC posiblemente **reversible** → Obtenemos los contenidos de la memoria *flash*

```
257 // calc crc
258 u16_t crc = 0xffff;
259 const u32_t start = SHMEM->user[BOOTLOADER_SHMEM_SPIF_FW_ADDR_UIX] + sizeof(fw_upgrade_info);
260 for (addr = start; addr < start + fui.len; addr++) {
261     u8_t c;
262     BL_IMG_READ(addr, (u8_t*)&c, 1);
263     crc = _bootloader_crc_ccitt_16(crc, c);
264 }
265
266 if (crc != fui.crc) {
267     b_putstr(" fw file bad crc:");
268     b_puthex16(crc);
269     b_put('/');
270     b_puthex16(fui.crc);
271     b_put('\n');
272     return FALSE;
273 }
```

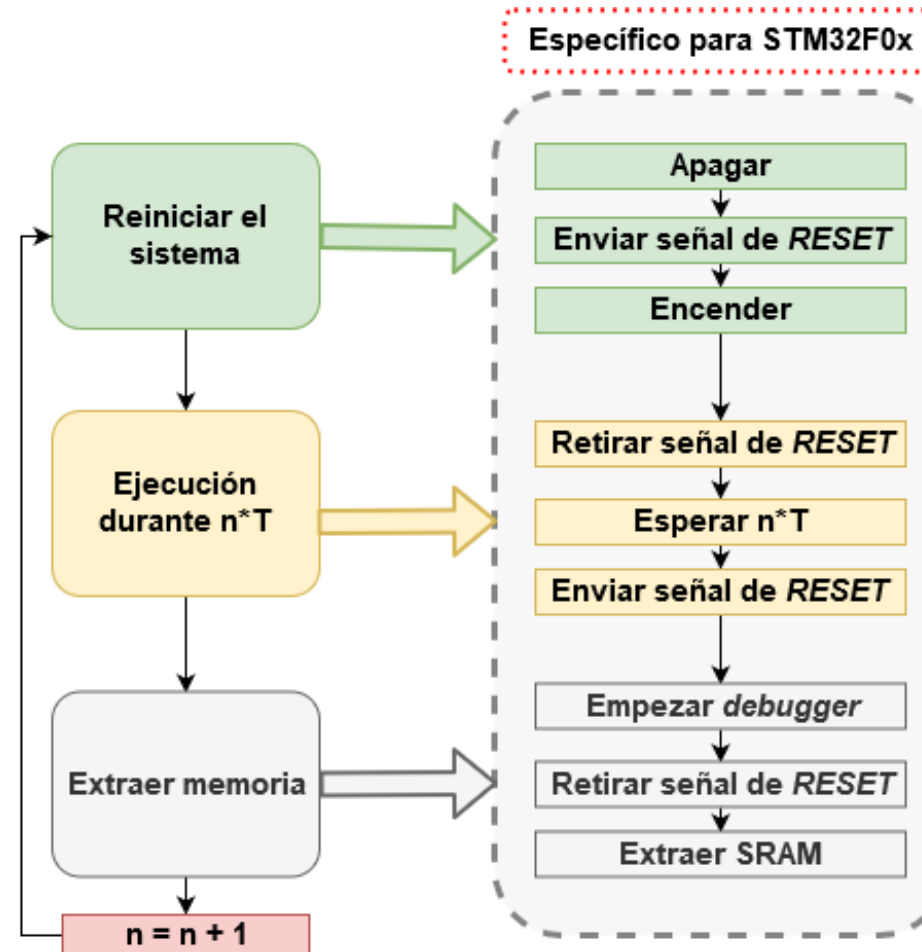


Loops over flash calculating CRC

3. Técnicas de Ataque a la Protección Contra Lectura

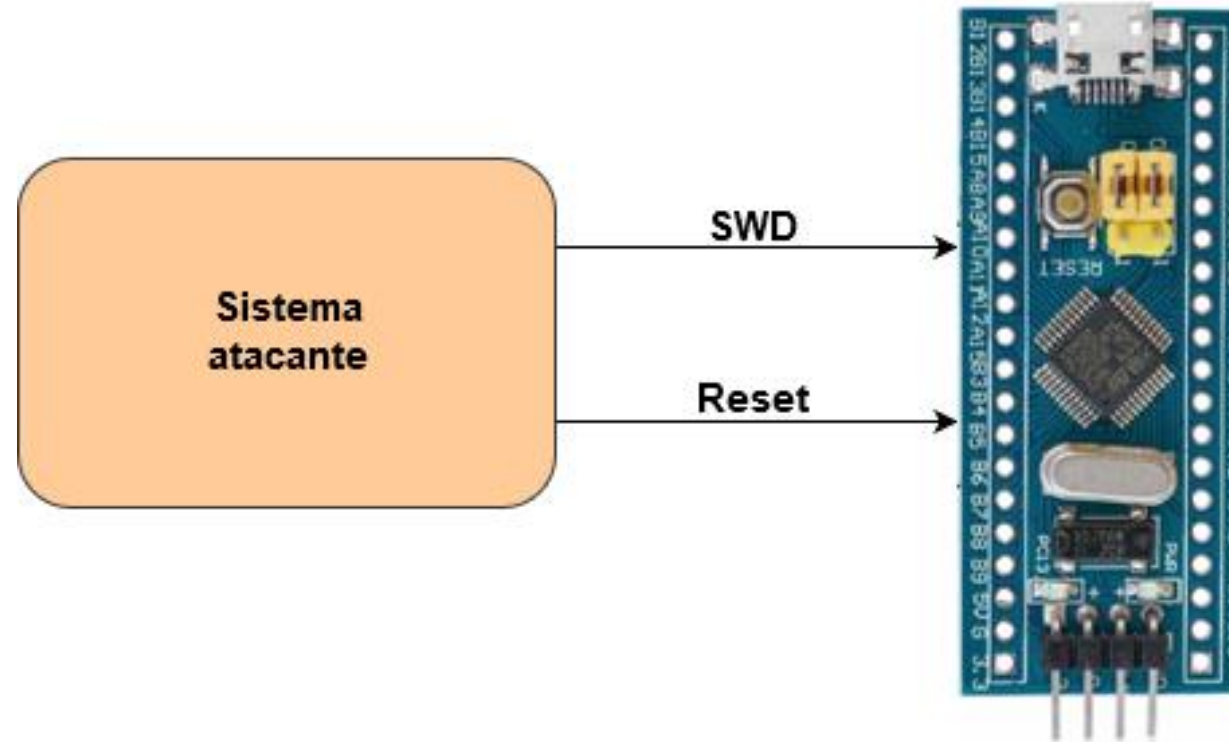
Cold Boot Stepping

- Cada iteración CRC tarda T segundos
- Poco nivel de control con pulsos de reloj rápidos
- Podría ser de utilidad un reloj externo



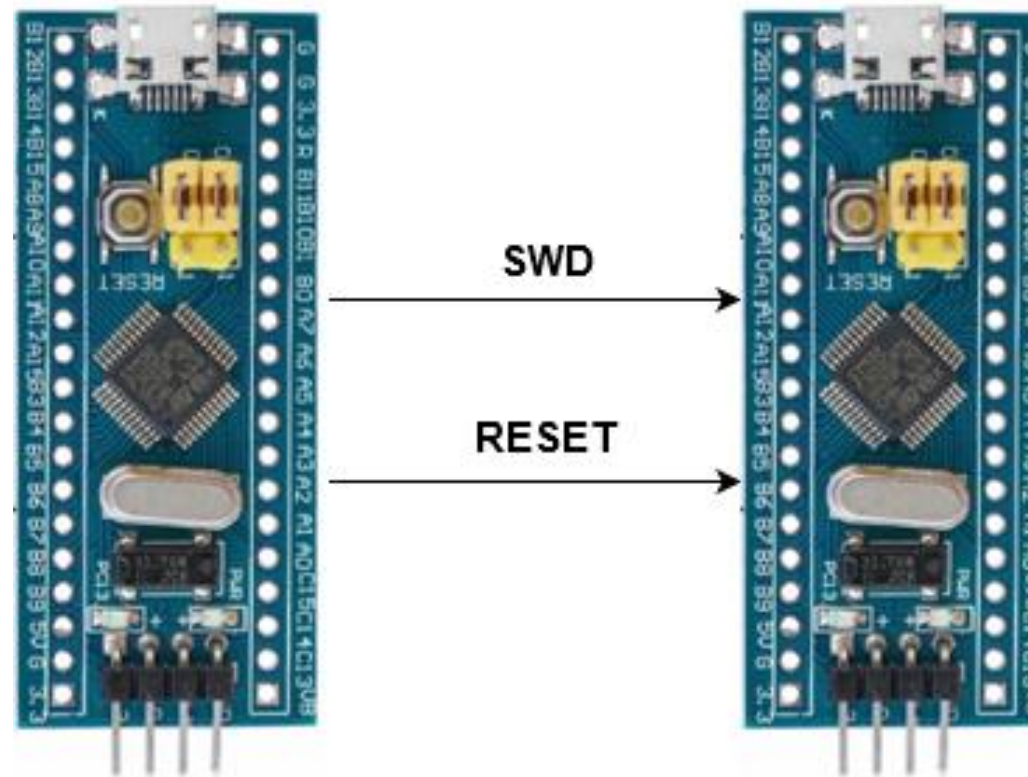
3. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping



3. Técnicas de Ataque a la Protección Contra Lectura

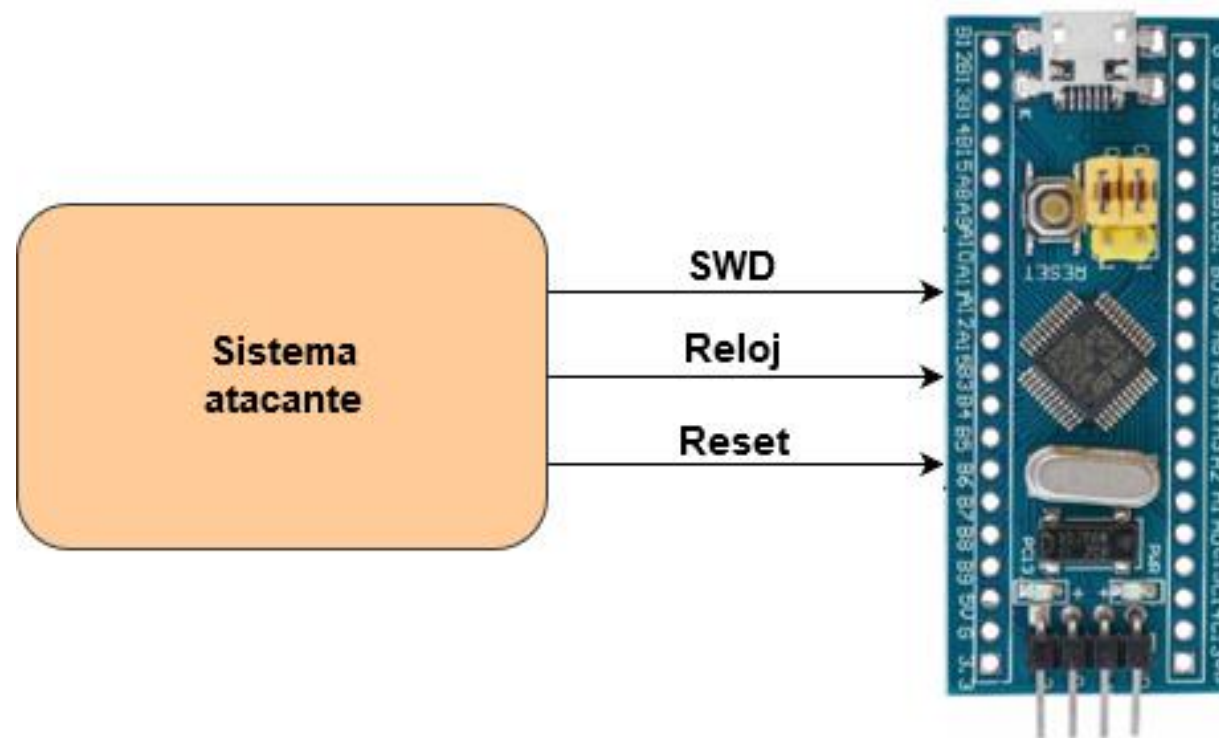
Cold Boot Stepping



3. Técnicas de Ataque a la Protección Contra Lectura

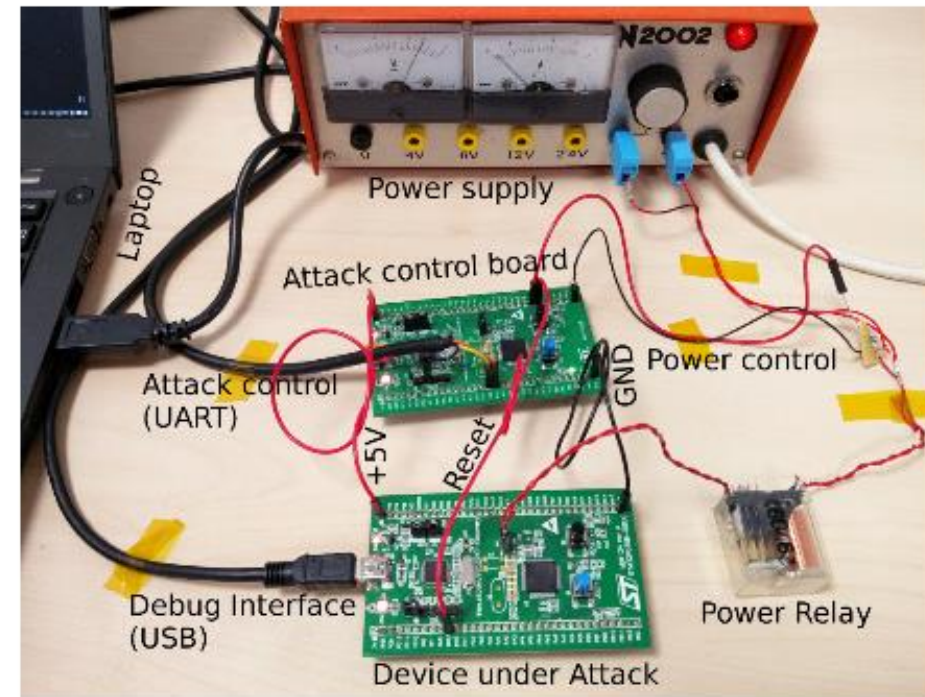
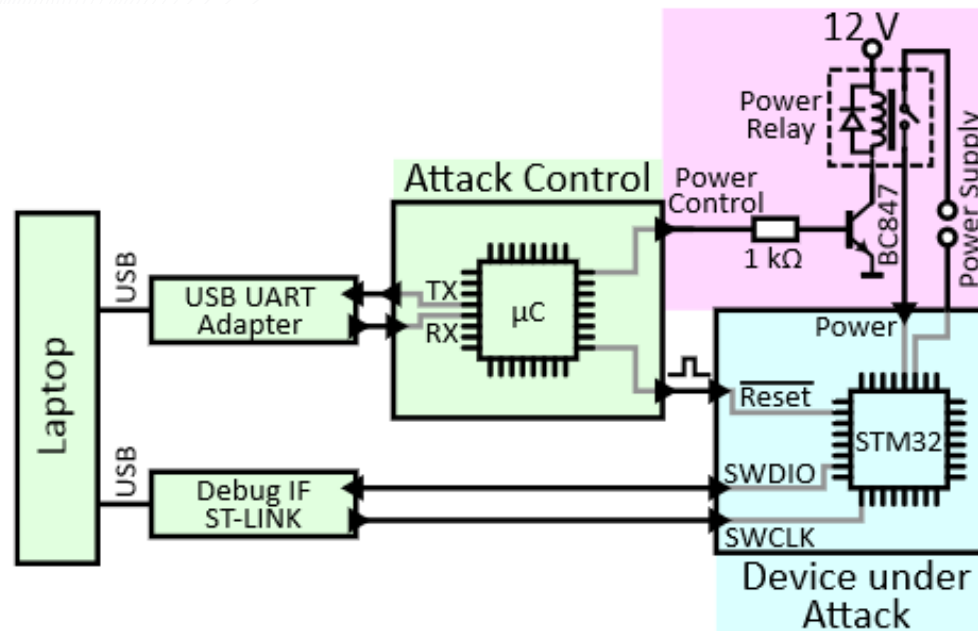
Cold Boot Stepping

- **Ralentizar** el reloj usando un oscilador externo
- No siempre es posible, depende del chip



3. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping



Fuente: <https://www.usenix.org/system/files/conference/woot17/woot17-paper-obermaier.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas

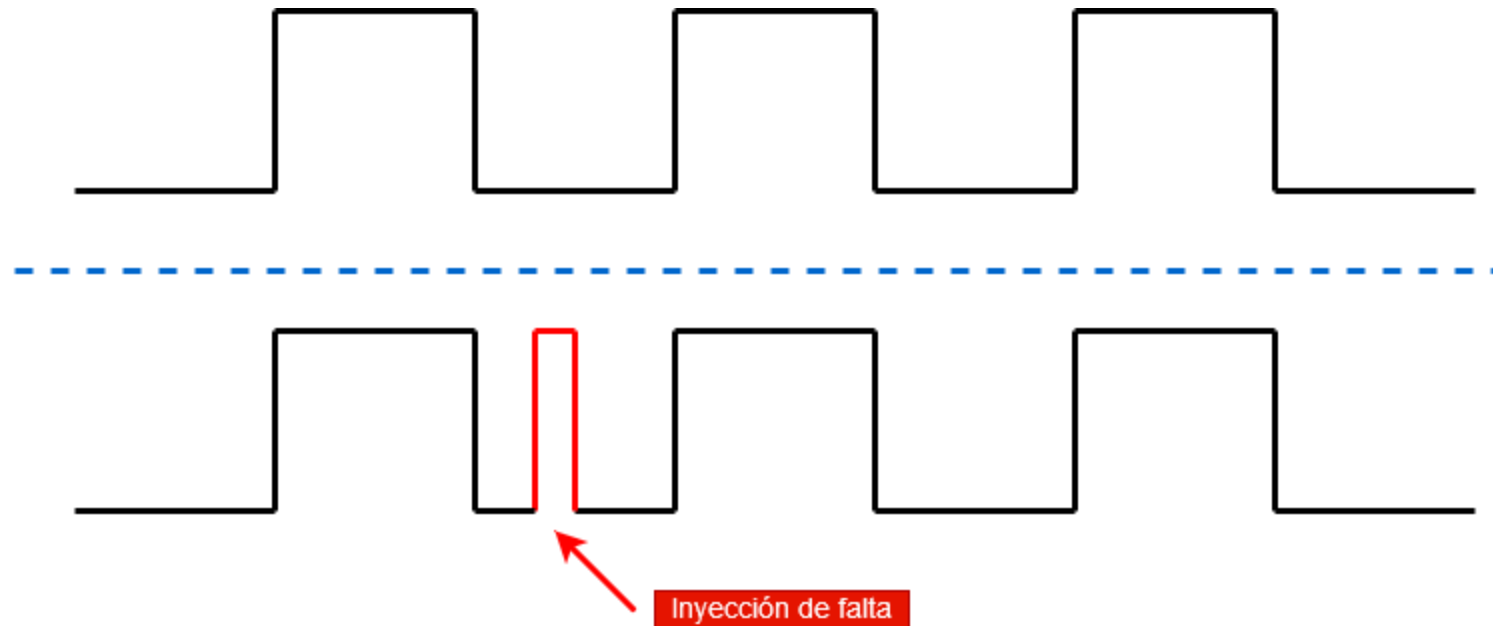
- El éxito y el tiempo que se necesita para llevar a cabo este ataque son poco predecibles
- El objetivo es enviar un pulso de reloj en un intervalo de tiempo no esperado, causando que se salten instrucciones



3. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas

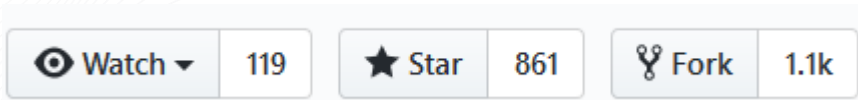
- El éxito y el tiempo que se necesita para llevar a cabo este ataque son poco predecibles
- El objetivo es enviar un pulso de reloj en un intervalo de tiempo no esperado, causando que se salten instrucciones



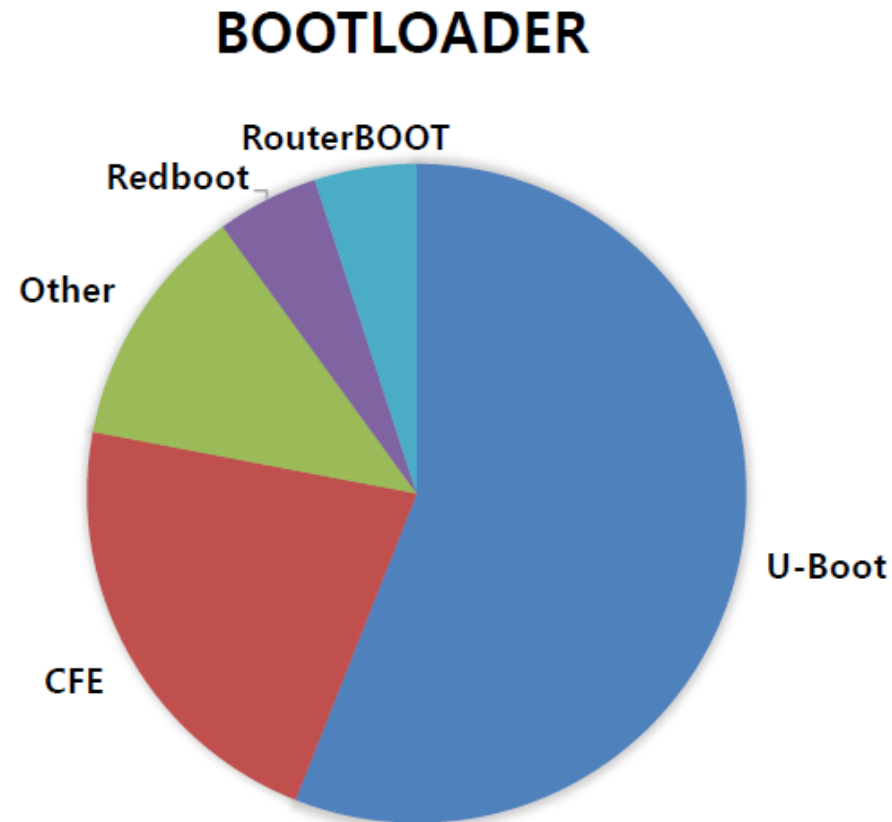
3. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas

- Existen diferentes *bootloaders*
- **U-Boot** es el más común



Fuente: <https://github.com/u-boot/u-boot>



Fuente: https://wikidevi.com/wiki/Property:Stock_bootloader/full

3. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas

- Si la función `autoboot_command` falla, obtenemos un **Shell**
- Hacemos que falle mediante **inyección de faltas**

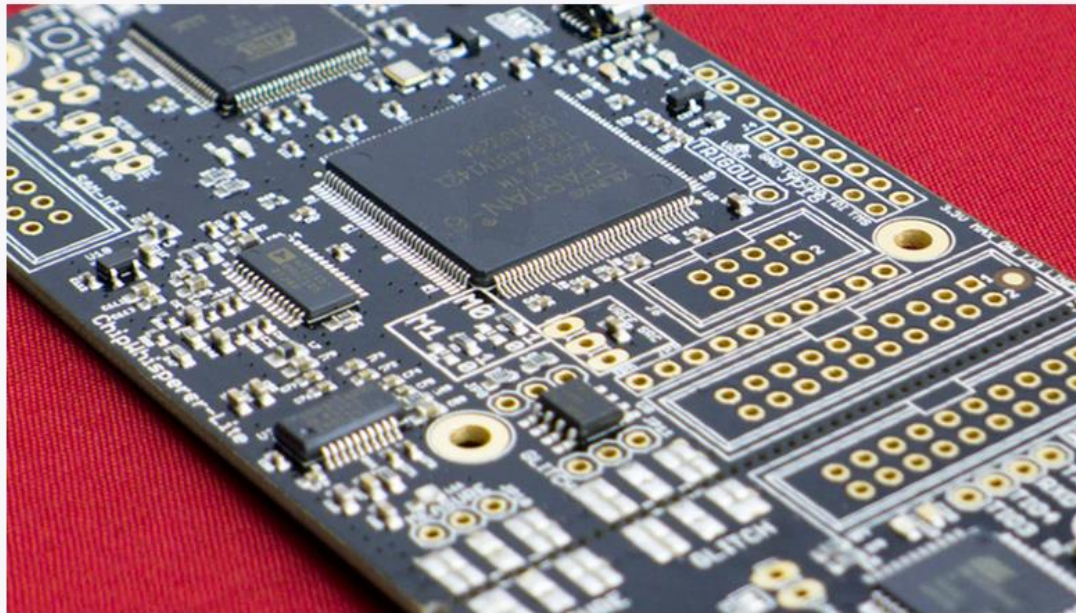
Fuente: <https://gitlab.denx.de/u-boot/u-boot/blob/master/common/main.c>

```
39  /* We come here after U-Boot is initialised and ready to process commands */
40  void main_loop(void)
41  {
42      const char *s;
43
44      bootstage_mark_name(BOOTSTAGE_ID_MAIN_LOOP, "main_loop");
45
46      if (IS_ENABLED(CONFIG_VERSION_VARIABLE))
47          env_set("ver", version_string); /* set version variable */
48
49      cli_init();
50
51      if (IS_ENABLED(CONFIG_USE_PREBOOT))
52          run_preboot_environment_command();
53
54      if (IS_ENABLED(CONFIG_UPDATE_TFTP))
55          update_tftp(0UL, NULL, NULL);
56
57      s = bootdelay_process();
58      if (cli_process_fdt(&s))
59          cli_secure_boot_cmd(s);
60
61      autoboot_command(s);
62
63      cli_loop();
64      panic("No CLI available");
65  }
```

3. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas

ChipWhisperer-Lite: A New Era of Hardware Security Research



Embedded security - is it an oxymoron? Learn the truth through a series of hands-on labs targeting computer and electrical engineers.

Created by

Colin O'Flynn

331 backers pledged CA\$ 88,535 to help bring this project to life.

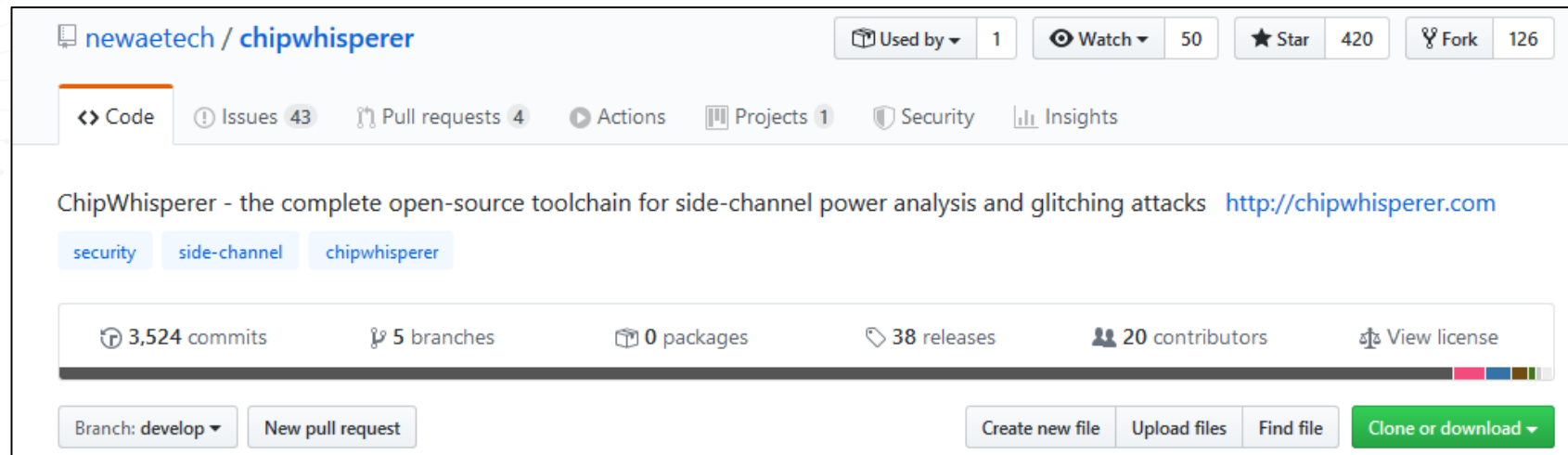
 **Last updated** July 5, 2016

Fuente: <https://www.kickstarter.com/projects/coflynn/chipwhisperer-lite-a-new-era-of-hardware-security>

3. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas

- Conjunto de herramientas para **la investigación de la seguridad** en sistemas empotrados
- Facilita la exploración de *side-channel attacks* e **inyecciones de falta**
- **Open Source**



Fuente: <https://github.com/newaetech/chipwhisperer>

3. Técnicas de Ataque a la Protección Contra Lectura

Side Channel Attack



מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE



DALHOUSIE
UNIVERSITY

IoT Goes Nuclear: Creating a ZigBee Chain Reaction

Eyal Ronen(✉)*, Colin O'Flynn†, Adi Shamir* and Achi-Or Weingarten*

**Weizmann Institute of Science, Rehovot, Israel*

{eyal.ronen, adi.shamir}@weizmann.ac.il

†Dalhousie University, Halifax, Canada

coflynn@dal.ca

Fuente: <https://eprint.iacr.org/2016/1047.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

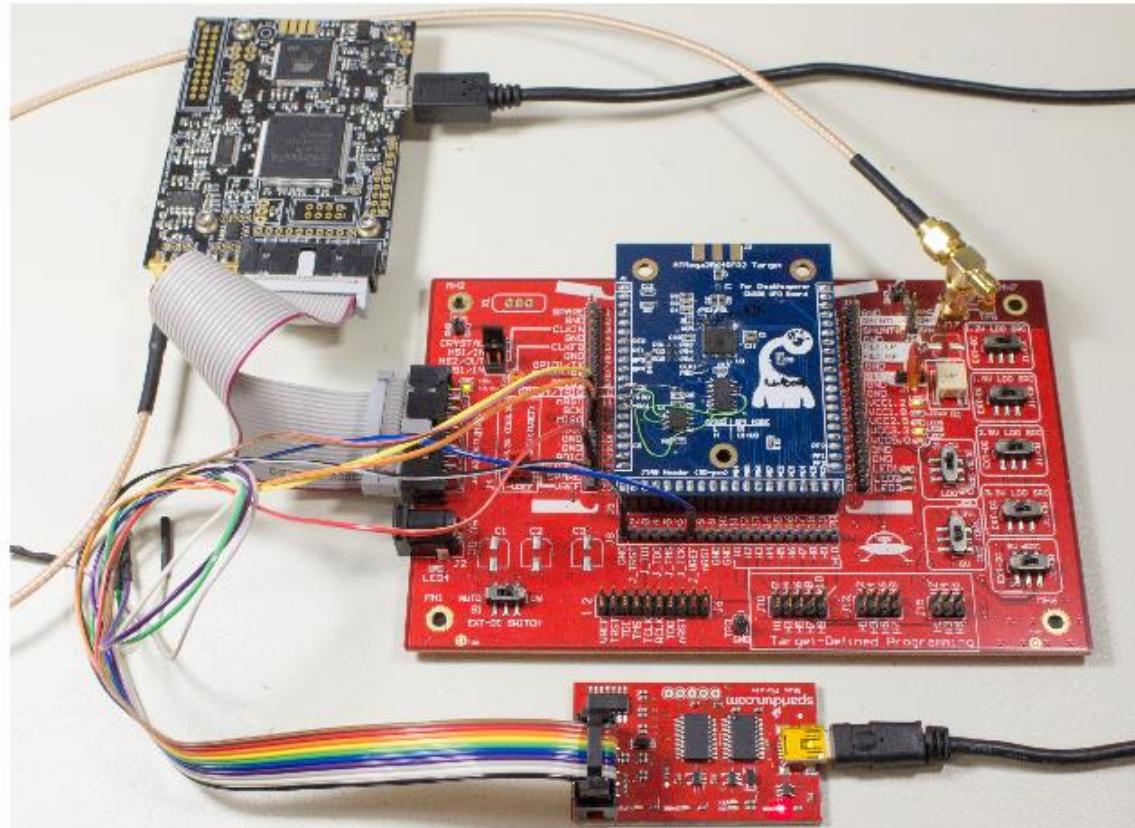
Side Channel Attack



Fuente: <https://eprint.iacr.org/2016/1047.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

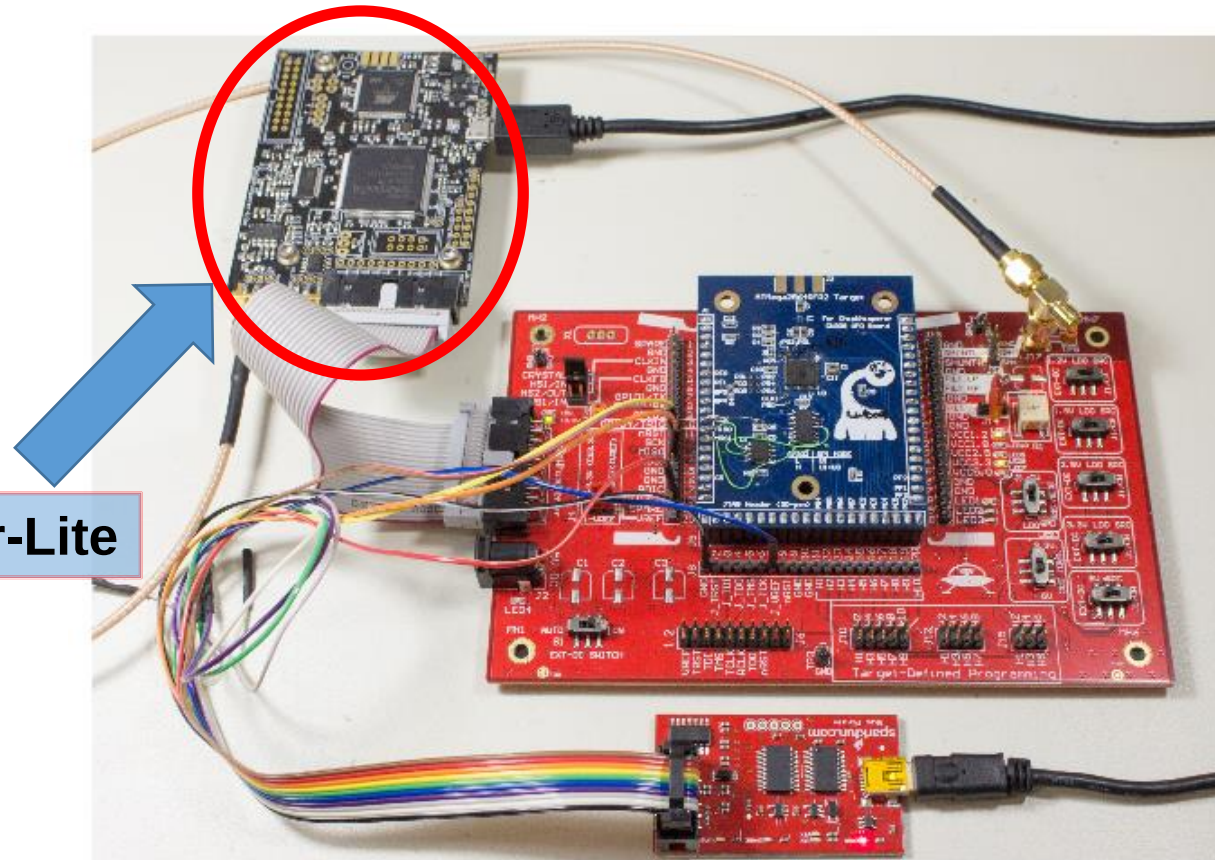
Side Channel Attack



Fuente: <https://eprint.iacr.org/2016/1047.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

Side Channel Attack



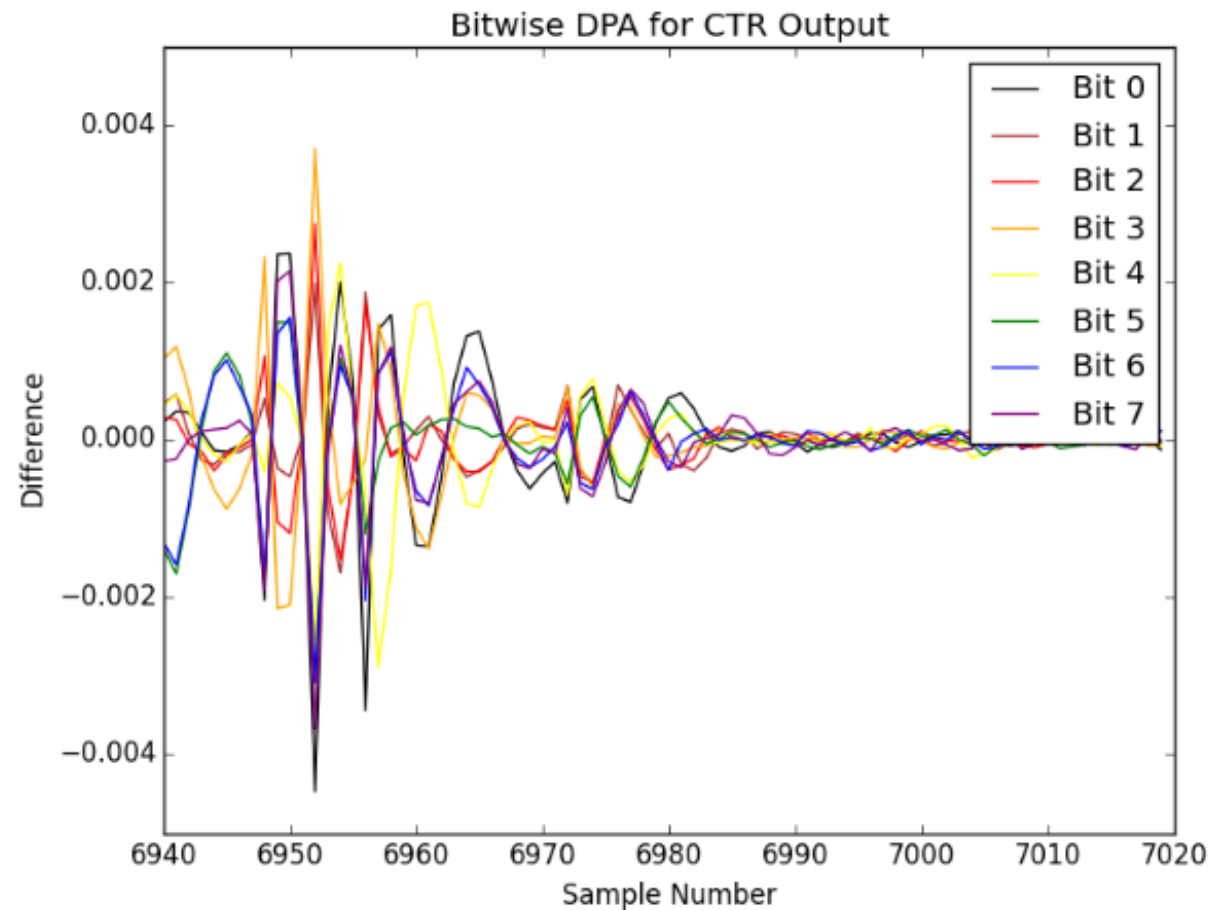
ChipWhisperer-Lite

Fuente: <https://eprint.iacr.org/2016/1047.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura

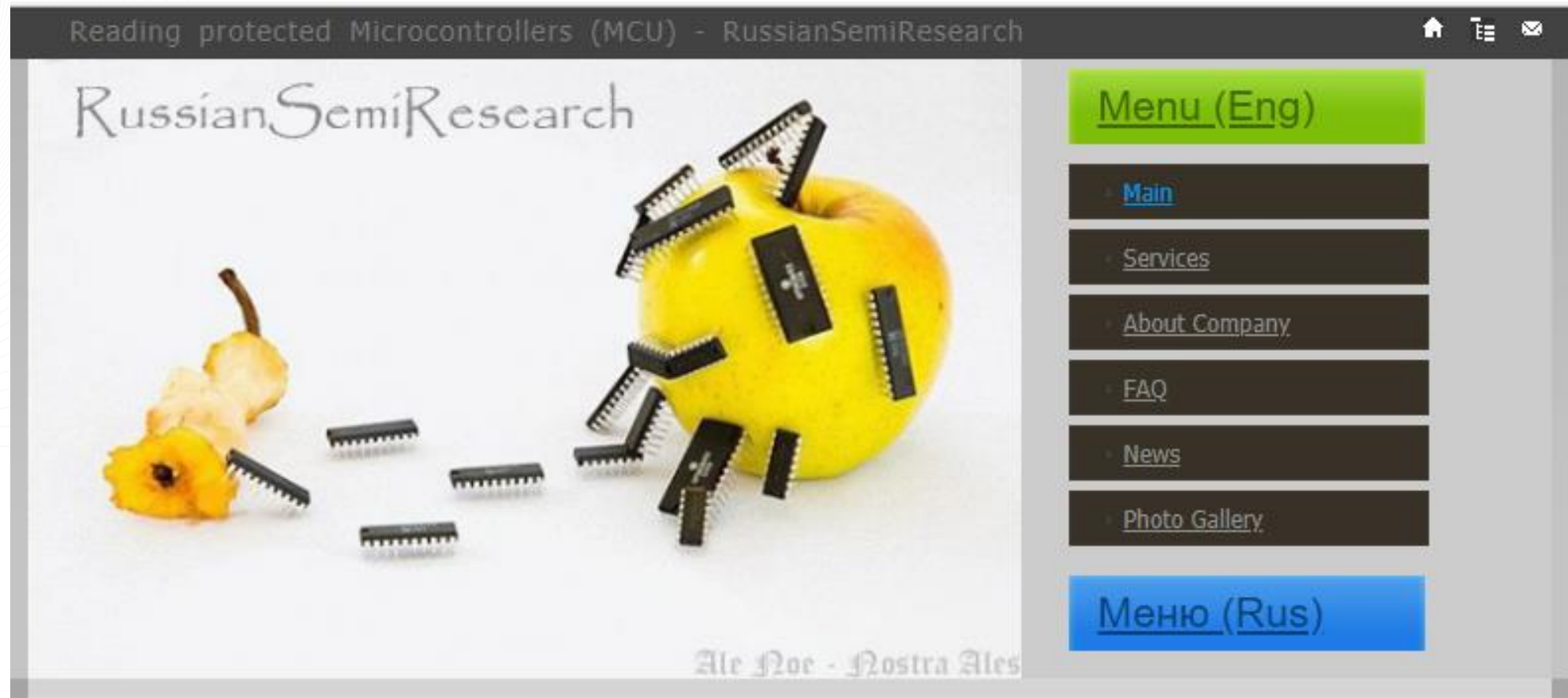
Side Channel Attack

- Permite inferir el valor de uno o varios bits
- Realizable con herramientas como: osciloscopio o **ChipWhisperer**
- Posible ataque: extraer las **claves de cifrado**



Fuente: <https://eprint.iacr.org/2016/1047.pdf>

3. Técnicas de Ataque a la Protección Contra Lectura



4. Conclusiones

- Normalmente no tenemos **acceso físico** a los dispositivos
- Este paradigma cambia con la introducción de las **Smart cities e IoT**
- Con acceso físico la **superficie de ataque** incrementa considerablemente
- Evitar la **seguridad por oscuridad**
- Diseñar el **firmware** asumiendo que la protección de lectura fallará
- Importancia de la **certificación** para los dispositivos IoT (Common Criteria, LINCE...)



COMUNIDAD Y CONFIANZA,
BASES DE NUESTRA CIBERSEGURIDAD

#XIIIJORNADASCCNCERT

OC.CCN.CNI.ES

WWW.CCN.CNI.ES

WWW.CCN-CERT.CNI.ES

**XIII
JORNADAS
STIC
CCN-CERT**

ccn-cert
centro criptológico nacional