

# Connected Factories lecture 5/6: Security

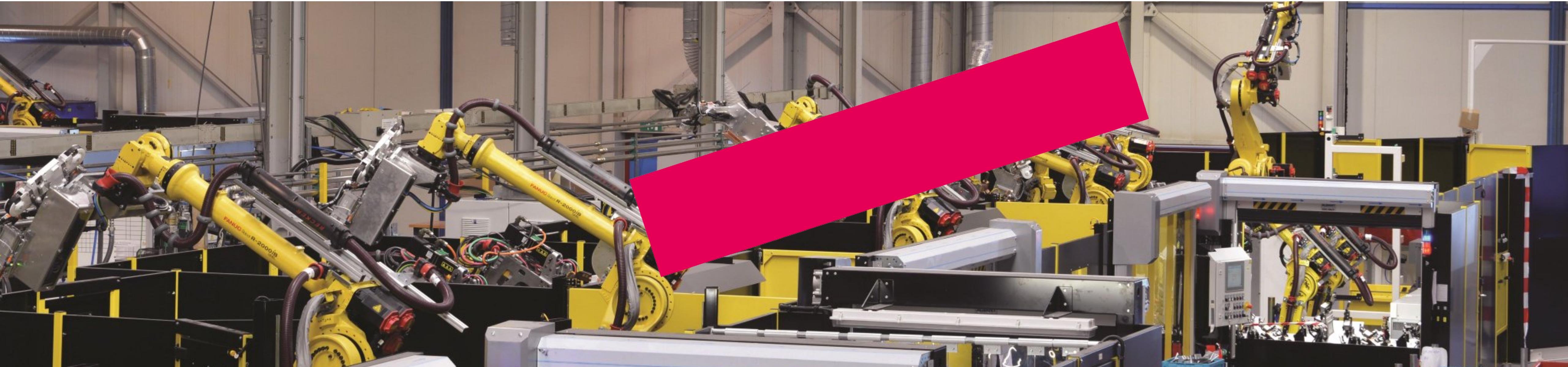


Image: AWL-Techniek

Associate Degree Smart Industry  
Faculty of Engineering and Automotive

johan.korten@han.nl

V1.0 May 26, 2020

# Schedule

	<b>Theme</b>	
Lecture 1	Introduction	
Lecture 2	Network connections	
Lecture 3	Network protocols	
Lecture 4	Interconnections	
Lecture 5	<b>Today: Security</b>	
Lecture 6	Safety	
Assessment		

# Security by design

Security (and safety) should be considered at design time already.

Patching leaks as we go (like the Microsoft approach) has proven to be a road to disaster.

How to enter a ‘secured’ area and wreak havoc:



Untapped Cities by Michelle Young

# Common IoT Security Issues: TEDx by Ken Munro

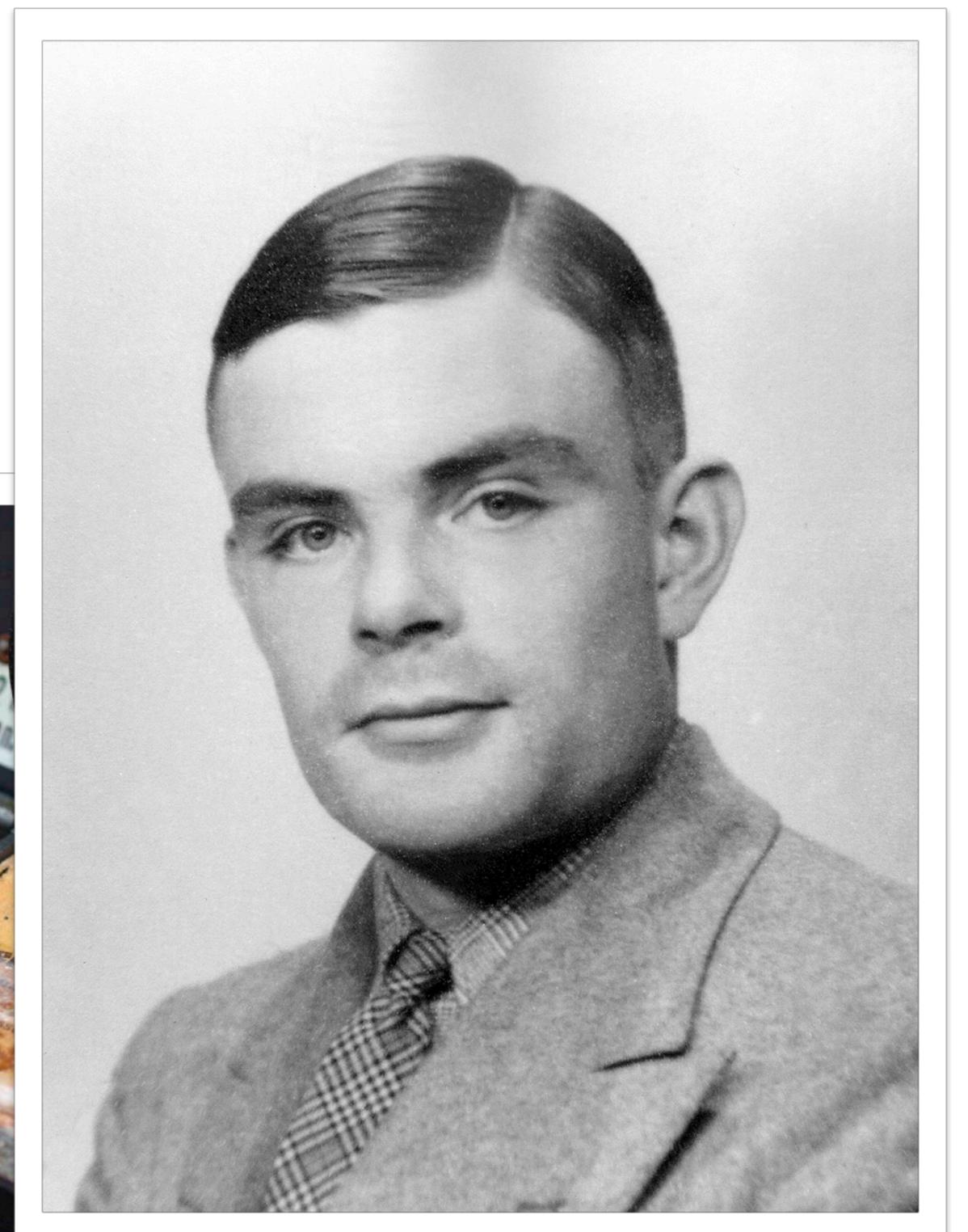
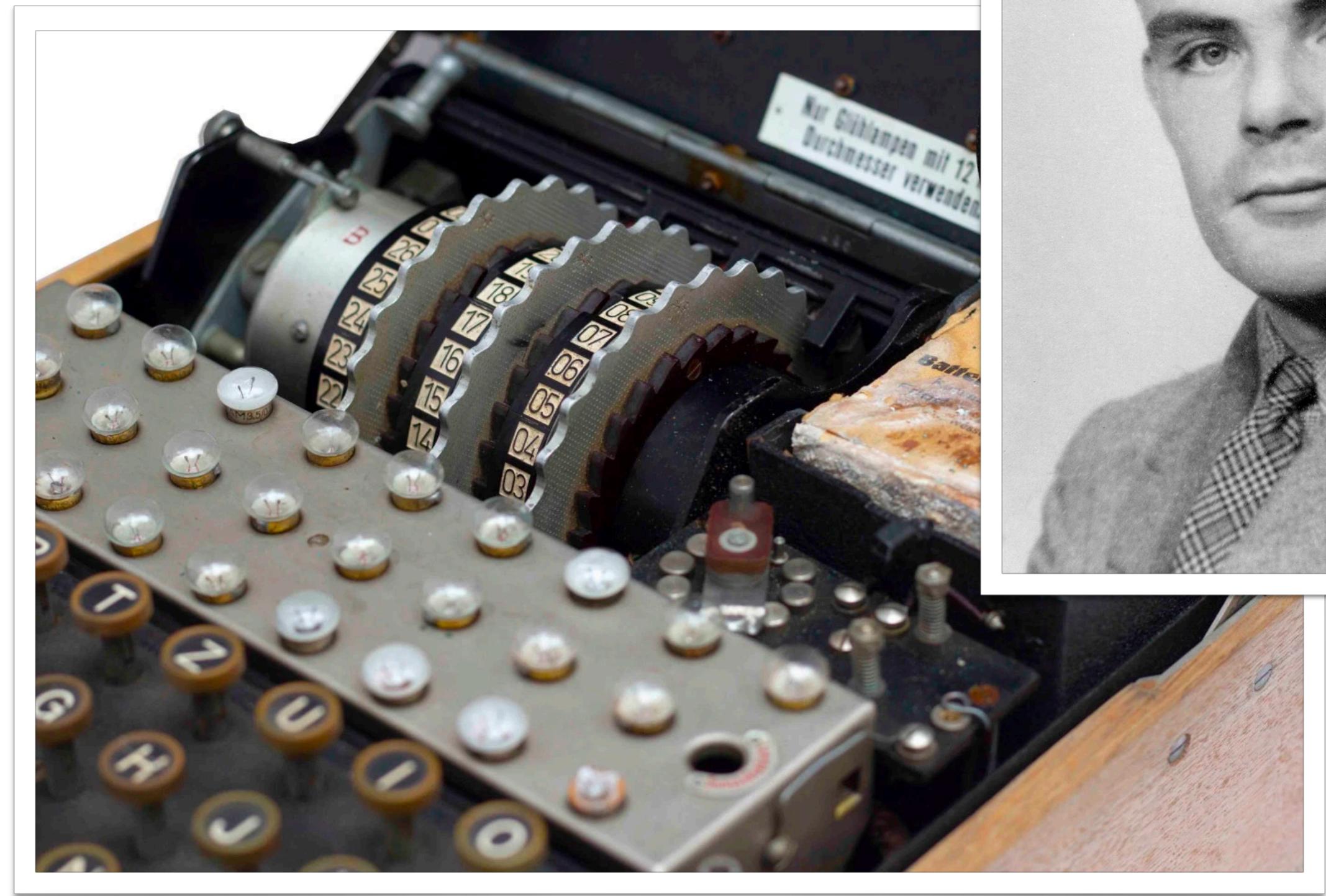
<https://youtu.be/pGtnC1jKpMg>



# Embedded Systems security

Different approaches:

- Secure by design
- Secure by default
- Secure by obscurity

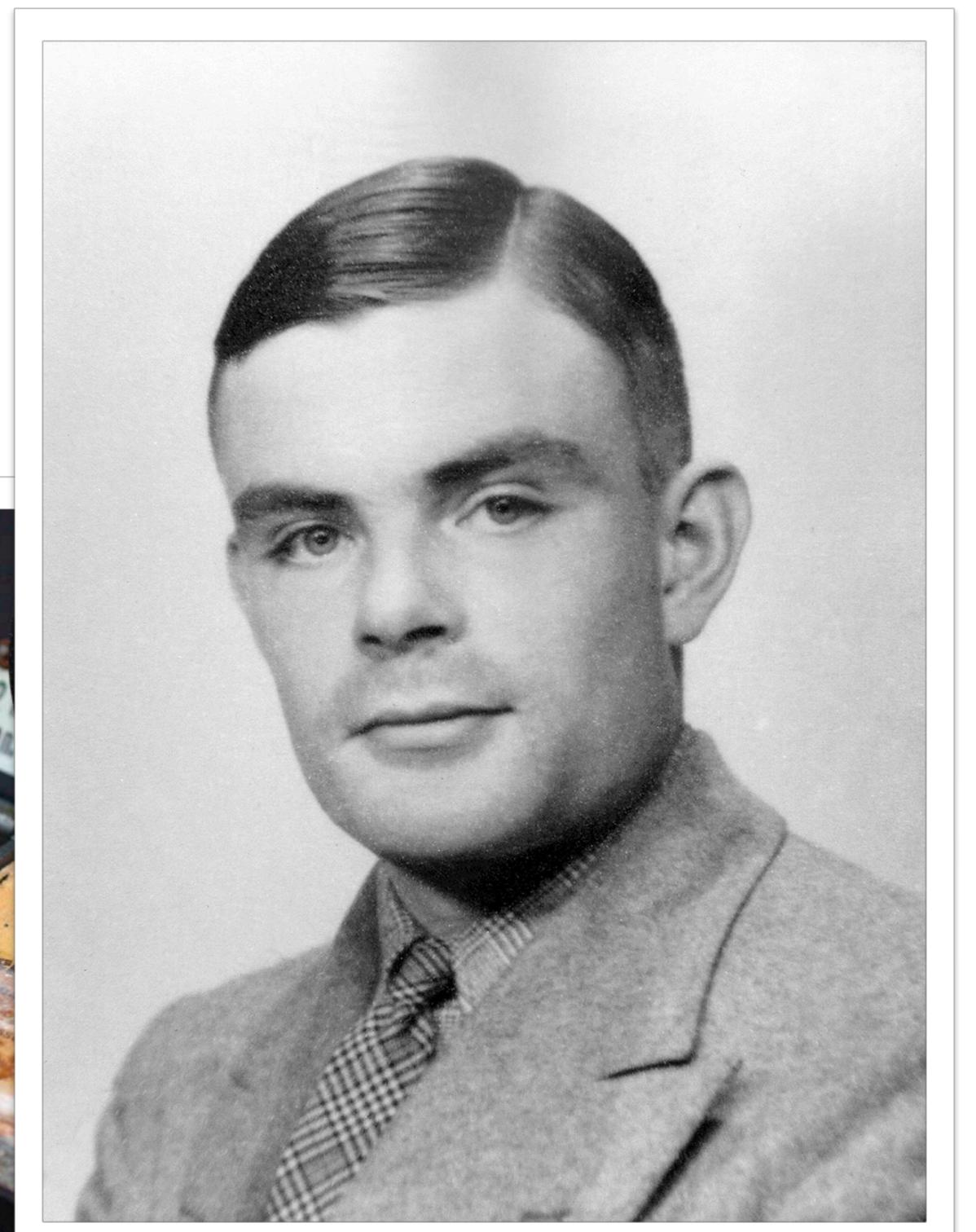
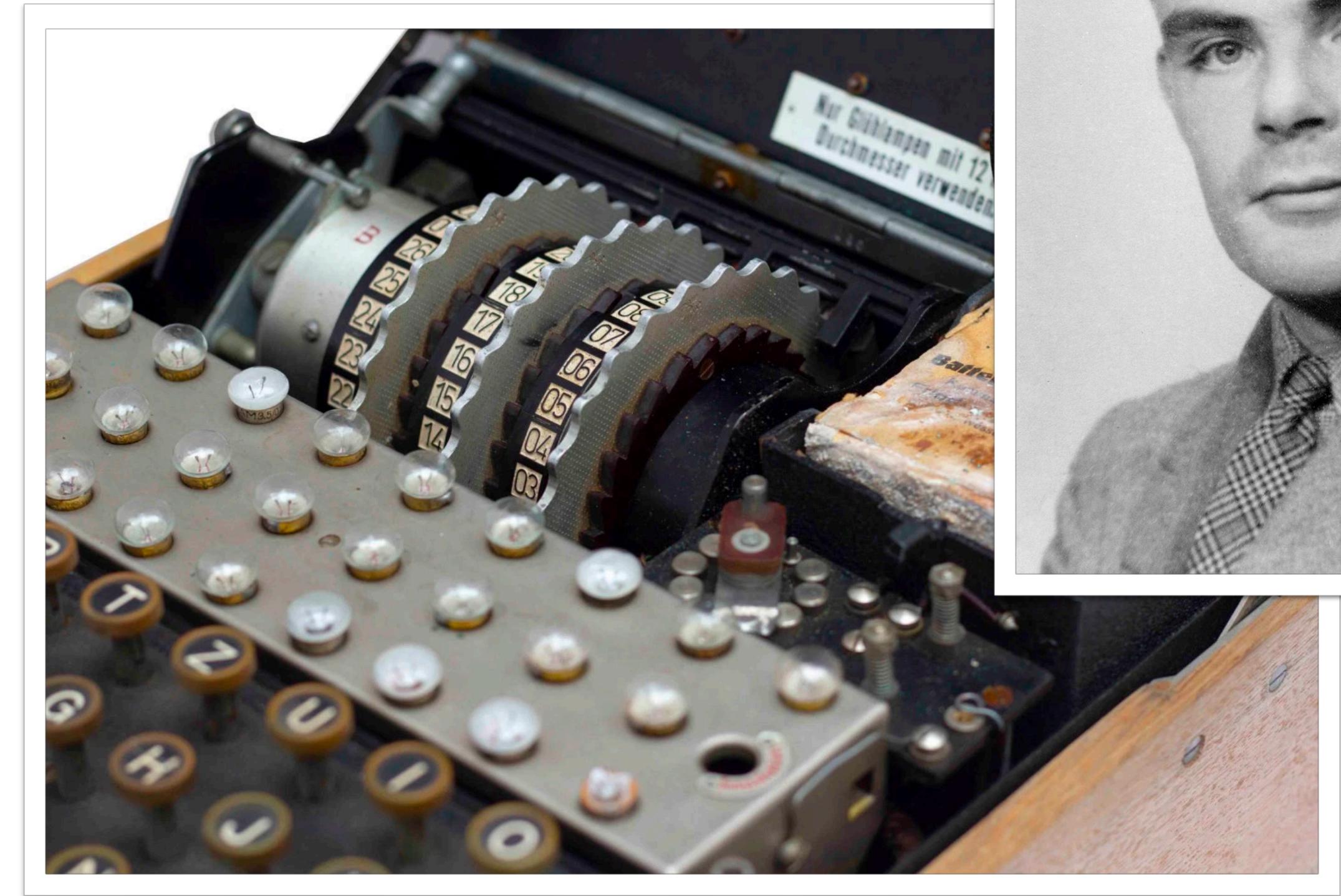


# How the allies cracked the enigma...

- Secure by obscurity:

*“reliance in security engineering on design or implementation secrecy as the main method of providing security to a system or component”*

[https://en.wikipedia.org/wiki/  
Security\\_through\\_obscurity](https://en.wikipedia.org/wiki/Security_through_obscurity)



# How the allies cracked the enigma...

- Secure by obscurity:

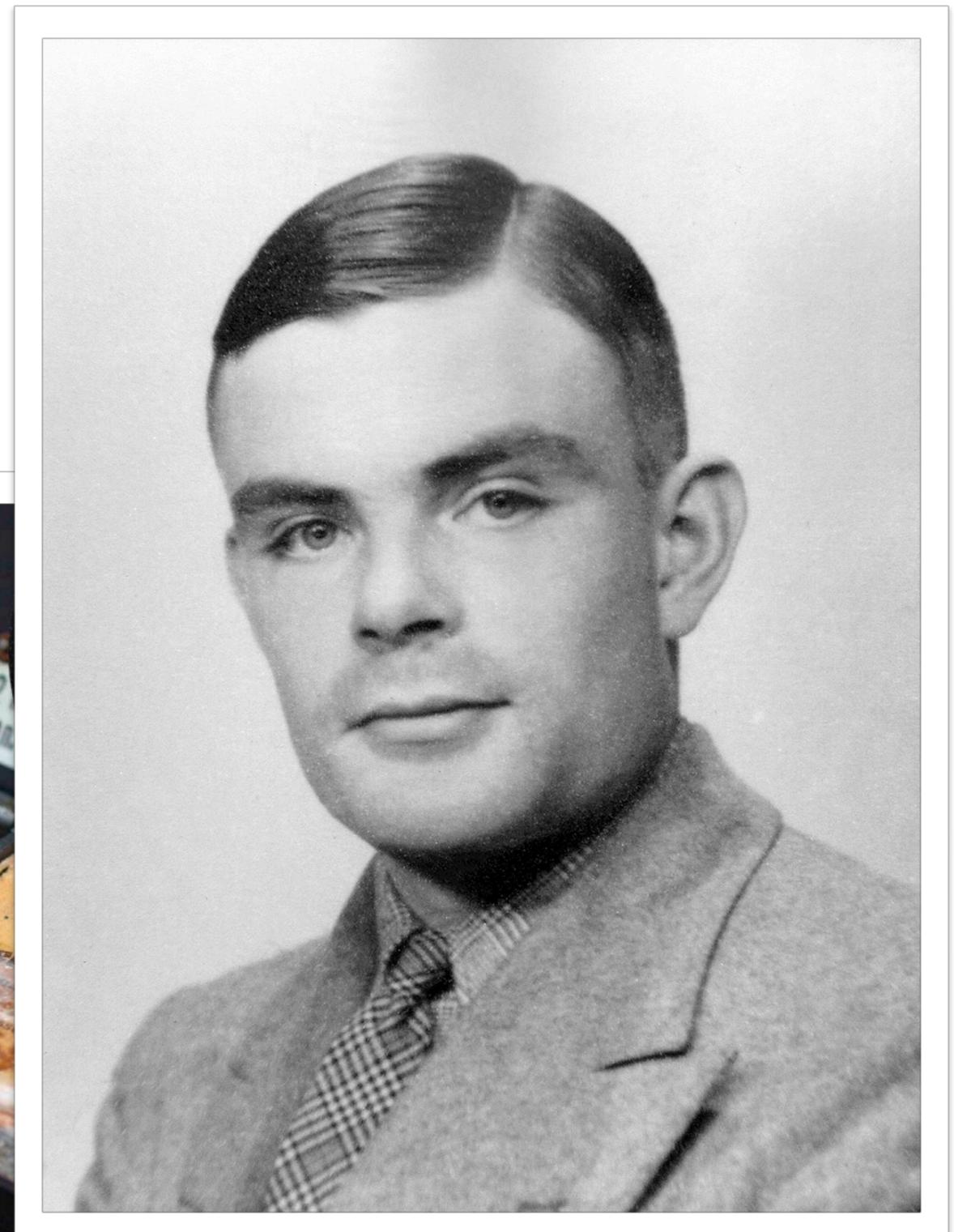
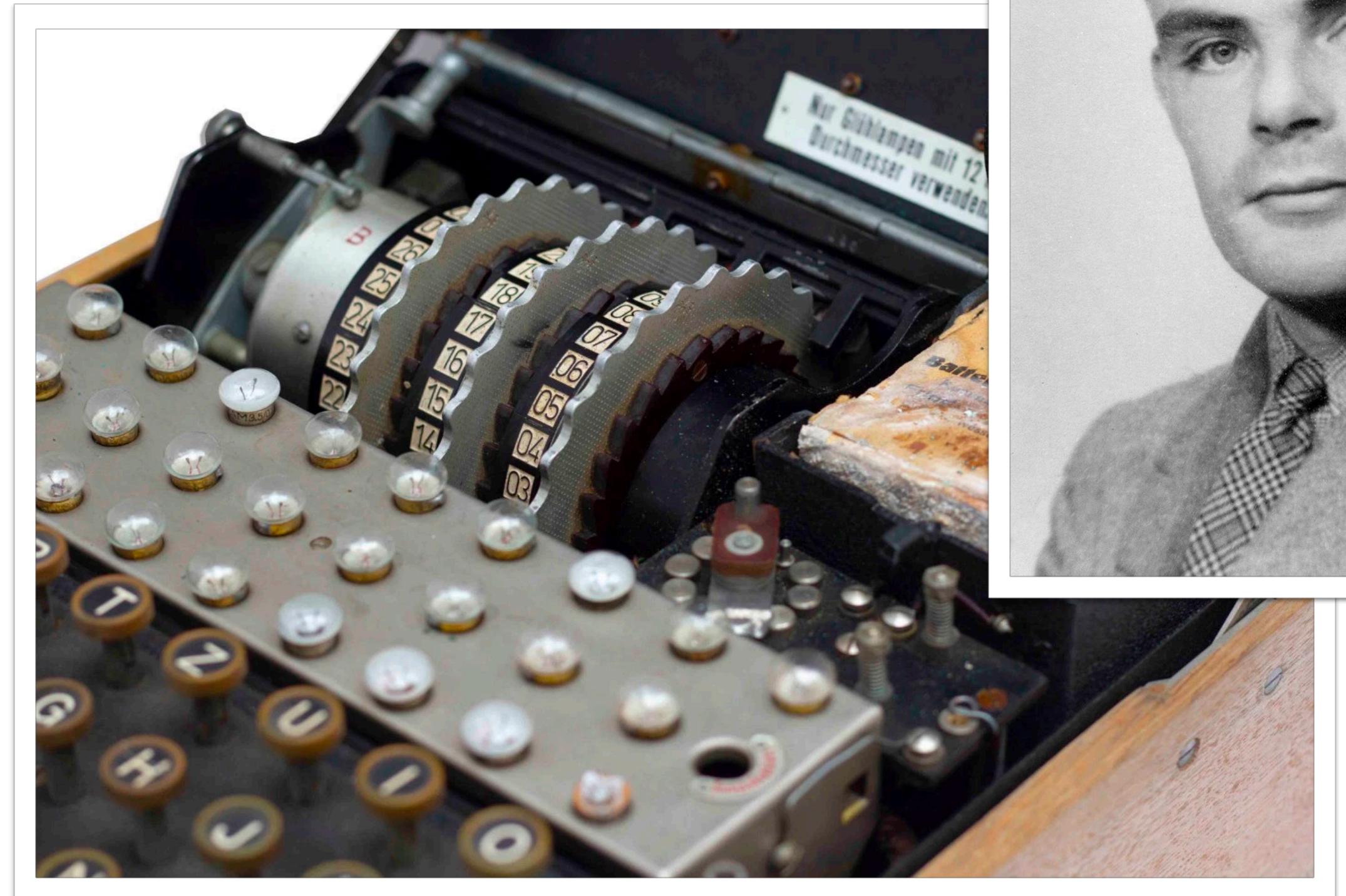
*"reliance in security engineering on design or implementation secrecy as the main method of providing security to a system or component"*

Initially (Original enigma) maybe..., but:

- Having the machine was not enough...
- Having a code book was not enough...

Even though...

- The Polish intelligence service was able to figure part of the machine out without ever having seen it.



# How the allies cracked the enigma...

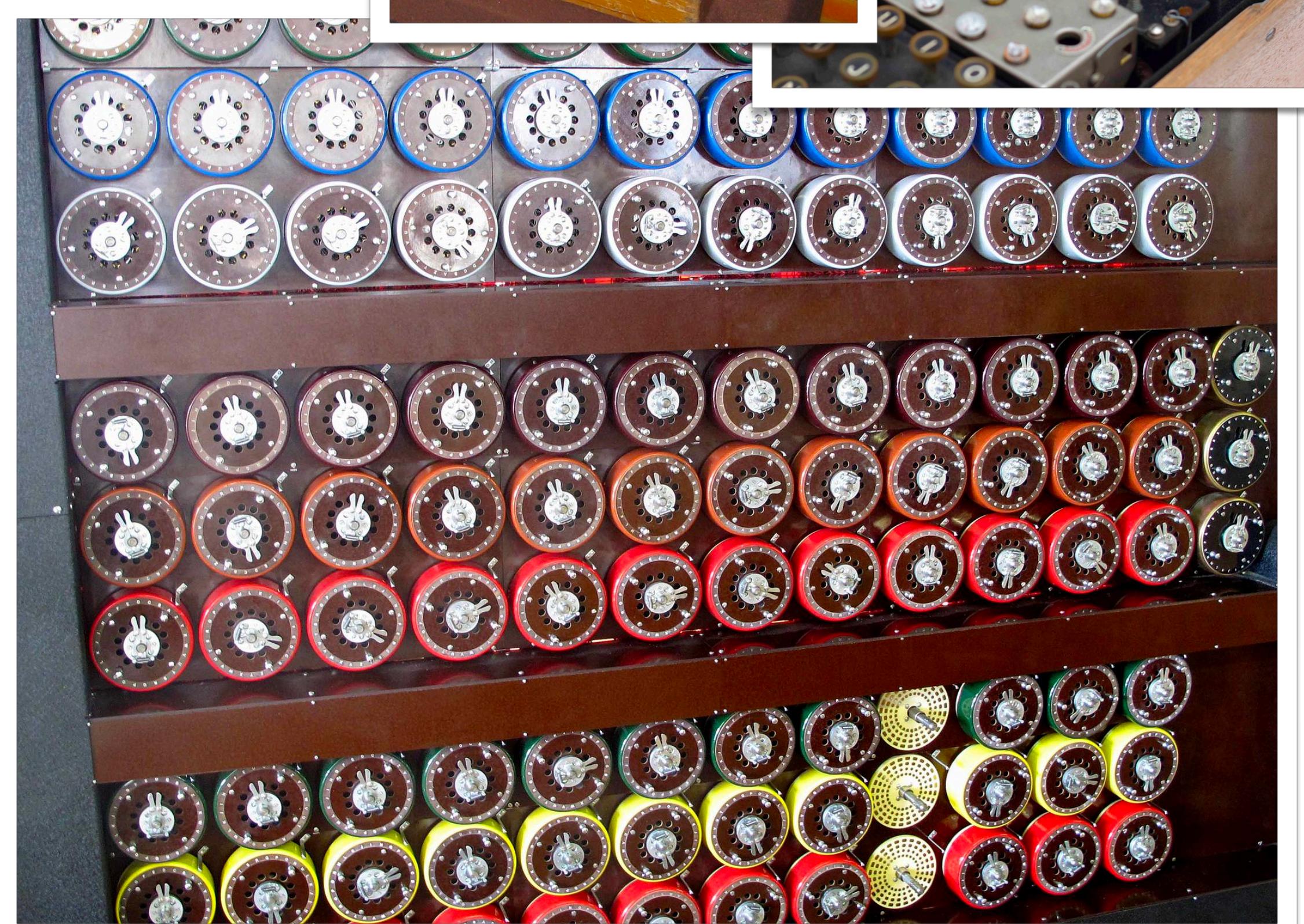
- The machine was quite “safe by design”:

158,962,555,217,826,360,000 combinations  
(Wehrmacht version)

- But it had some design flaws:

A letter could not be the same letter once encrypted.

Biggest flaw: not too robust against predictability of users.



# How the allies cracked the enigma...

A letter could not be the same letter once encrypted.

Biggest flaw: [not too robust against predictability of users.](#)

"The one snag with Enigma of course is the fact that if you press A, you can get every other letter but A. I picked up this message and—one was so used to looking at things and making instant decisions—I thought: 'Something's gone. What has this chap done? There is not a single L in this message.' My chap had been told to send out a dummy message and he had just had a fag [cigarette] and pressed the last key on the keyboard, the L. So that was the only letter that didn't come out. We had got the biggest crib we ever had, the encipherment was LLLL, right through the message and that gave us the new wiring for the wheel [rotor]. That's the sort of thing we were trained to do. Instinctively look for something that had gone wrong or someone who had done something silly and torn up the rule book. [\[103\]](#)"



# Common Networking Tools 1/2

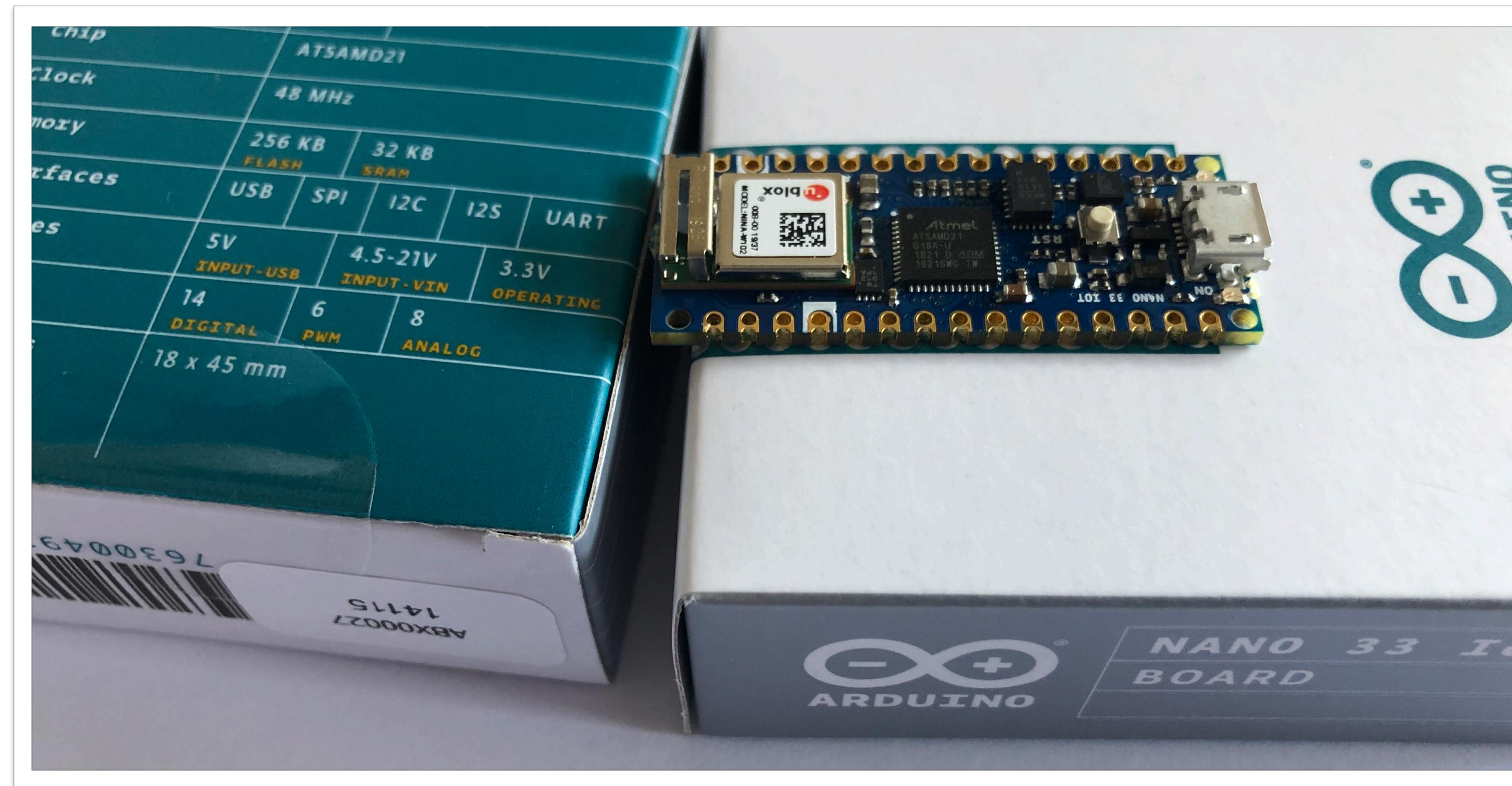
Tool name	Usage	Common Flag(s) / usage
netstat	Shows incoming and outgoing TCP connections	-r (ip route), -i (interfaces)
lsof	list open files	-i -n -P   grep programname (e.g. "lsof -i -n -P   grep mosquitto")
tcpdump	(TCP) packet analyser	
ping	Sends echo request to specified host	e.g. "ping www.han.nl"
traceroute	Shows route of a packet	-i (e.g. "traceroute -i google.com")
nmap	Network exploration and security scanner tool	

# Common Networking Tools 2/2

Tool name	Usage	Common Flag(s) / usage
<a href="#">dig</a> / <a href="#">nslookup</a> / <a href="#">host</a>	Show DNS record information	<a href="#">dig</a> / <a href="#">nslookup</a> / <a href="#">host</a>
<a href="#">hostname</a>	Returns hostname of your system	<a href="#">hostname</a>
<a href="#">ifconfig</a>	Network interface configuration overview tool	
<a href="#">whois</a>	Retrieving domain name information	
<a href="#">arp</a>	Display the ARP table	-an
<a href="#">ssh</a>	SSH client to connect to some host	ssh -l pi other_hostname

# Your secure IoT / MQTT device.

Your Nano 33 IoT Board also features a Crypto chip (ATECC608A)

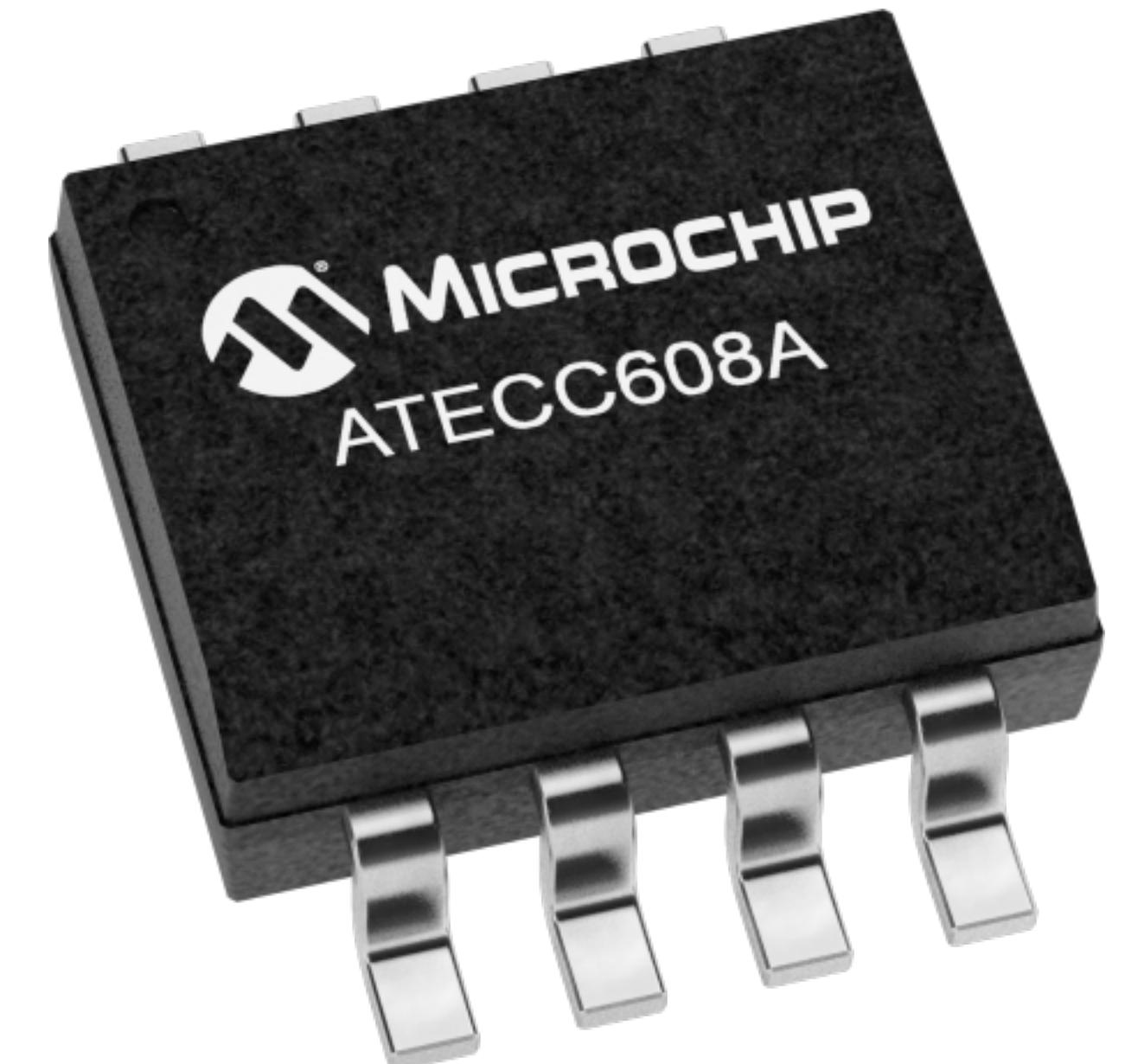


# Your secure IoT / MQTT device.

Microchip ATECC608A

## Features:

- Cryptographic co-processor with secure hardware-based key storage
- Protected storage for up to 16 Keys, certificates or data
- ECDH: FIPS SP800-56A Elliptic Curve Diffie-Hellman
- NIST standard P256 elliptic curve support
- SHA-256 & HMAC hash including off-chip context save/restore
- AES-128: encrypt/decrypt, galois field multiply for GCM



See also: <https://github.com/MicrochipTech/gcp-iot-core-examples/wiki>

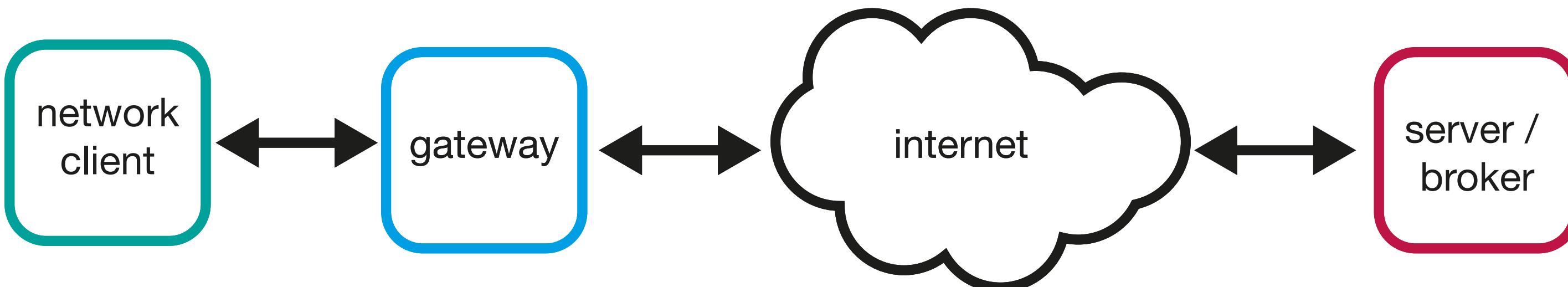
# ARP Poisoning / Spoofing

Why:

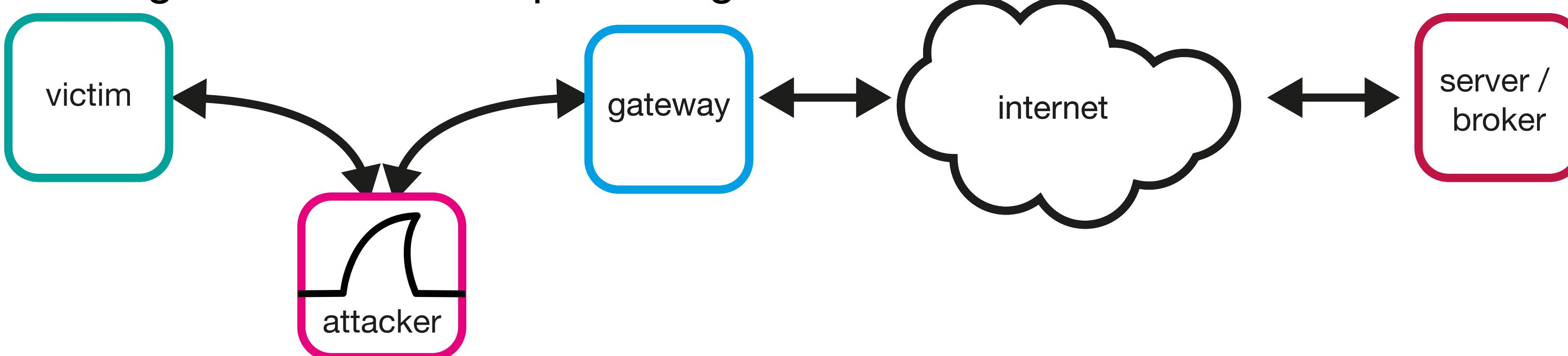
- get more insight into networking
- see why encryption matters
- get some intuition on how security can be tested

# How does ARP Poisoning / Spoofing work?

Normal routing operation:



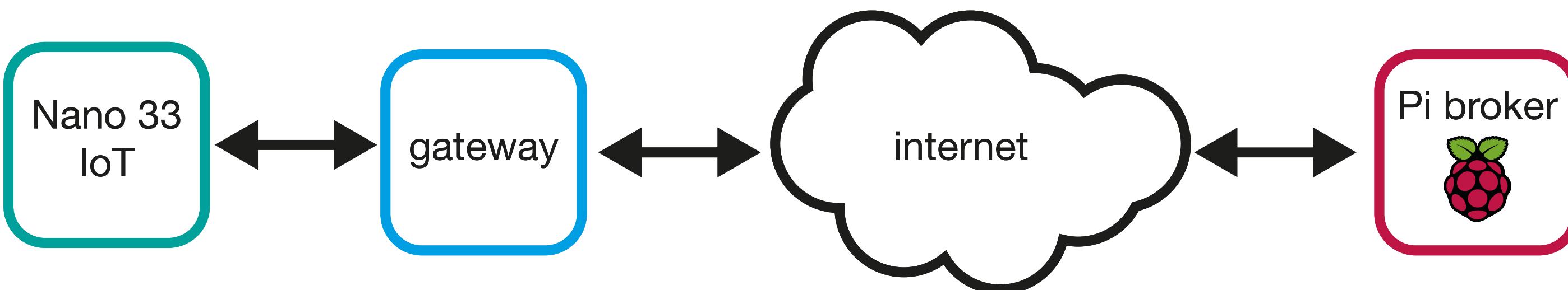
Routing with ARP cache poisoning:



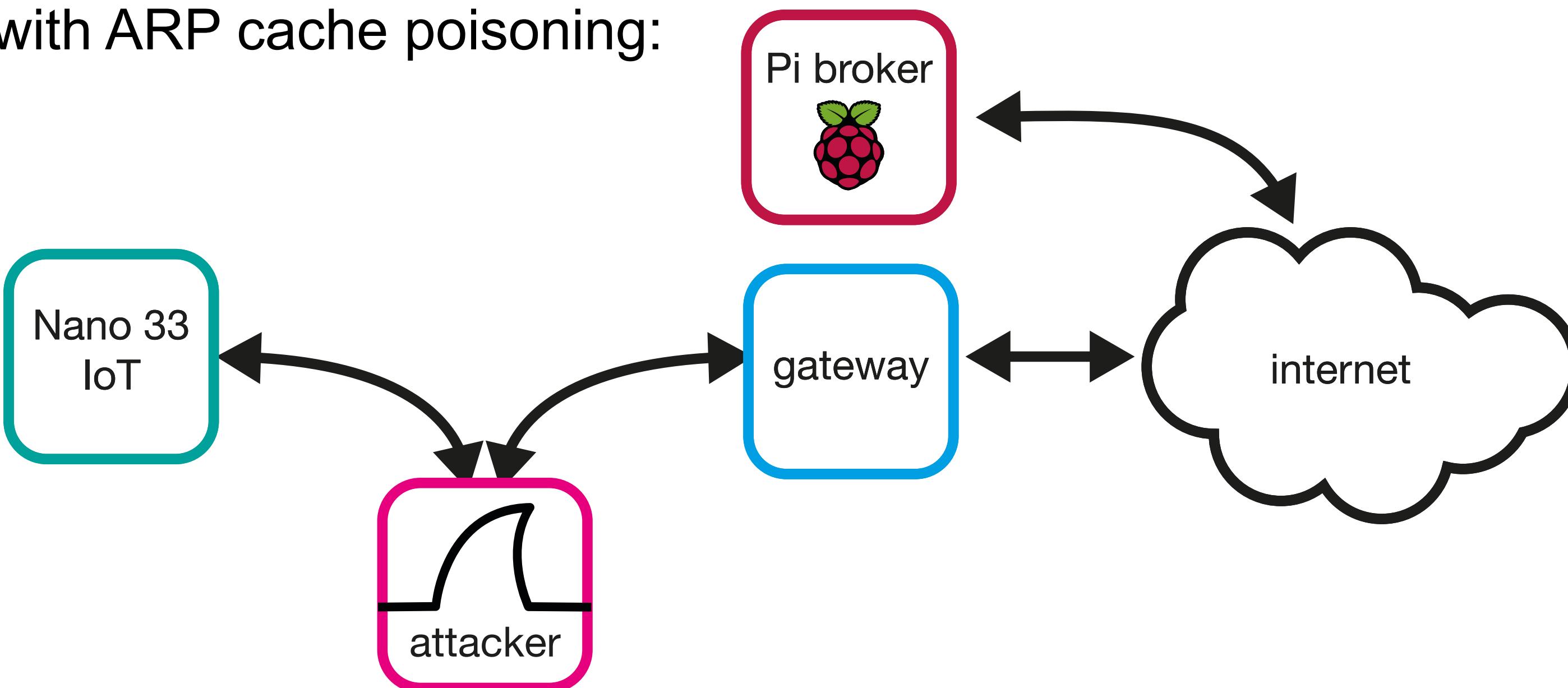
Note: victim needs to be in same subnetwork (wired, wireless, bridged)

# Ok, time for our very evil plans:

Normal routing operation:



Routing with ARP cache poisoning:



# Preparation: moving the broker outside of your local network

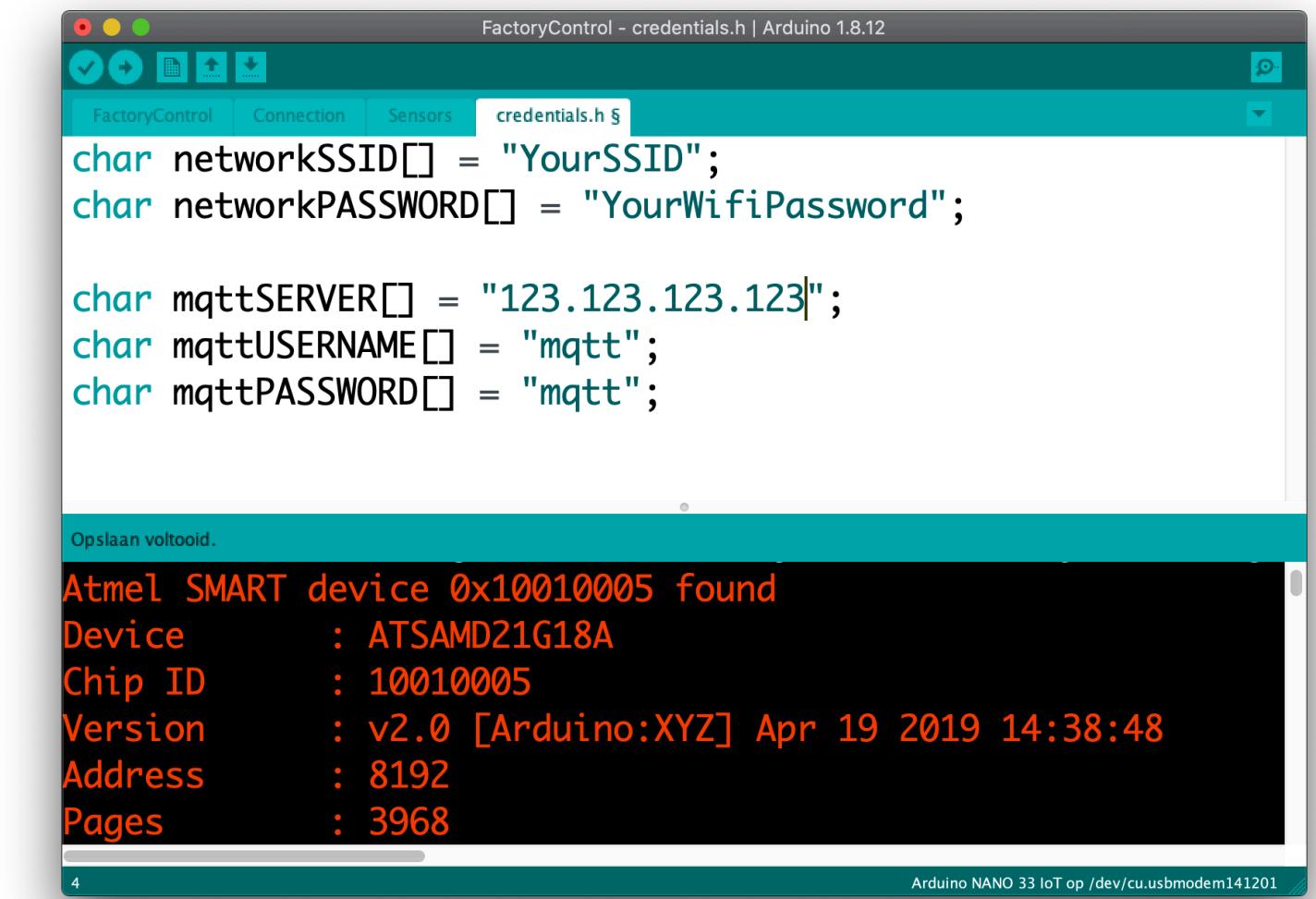
Either:

- Put your Pi in the DMZ of your Router (*Warning: this possibly puts your device at risk!*)

Or:

- Port Forward 1883 in your router to the local IP address of your Pi
- Change FactoryControl Arduino sketch credentials to (external) IP address of your ISP router

(Find it in your router settings or use <https://www.whatismyip.com>)



The screenshot shows the Arduino IDE interface with the following details:

- Sketch:** FactoryControl - credentials.h | Arduino 1.8.12
- Code (credentials.h):**

```
char networkSSID[] = "YourSSID";
char networkPASSWORD[] = "YourWifiPassword";

char mqttSERVER[] = "123.123.123.123";
char mqttUSERNAME[] = "mqtt";
char mqttPASSWORD[] = "mqtt";
```

- Terminal Output:**

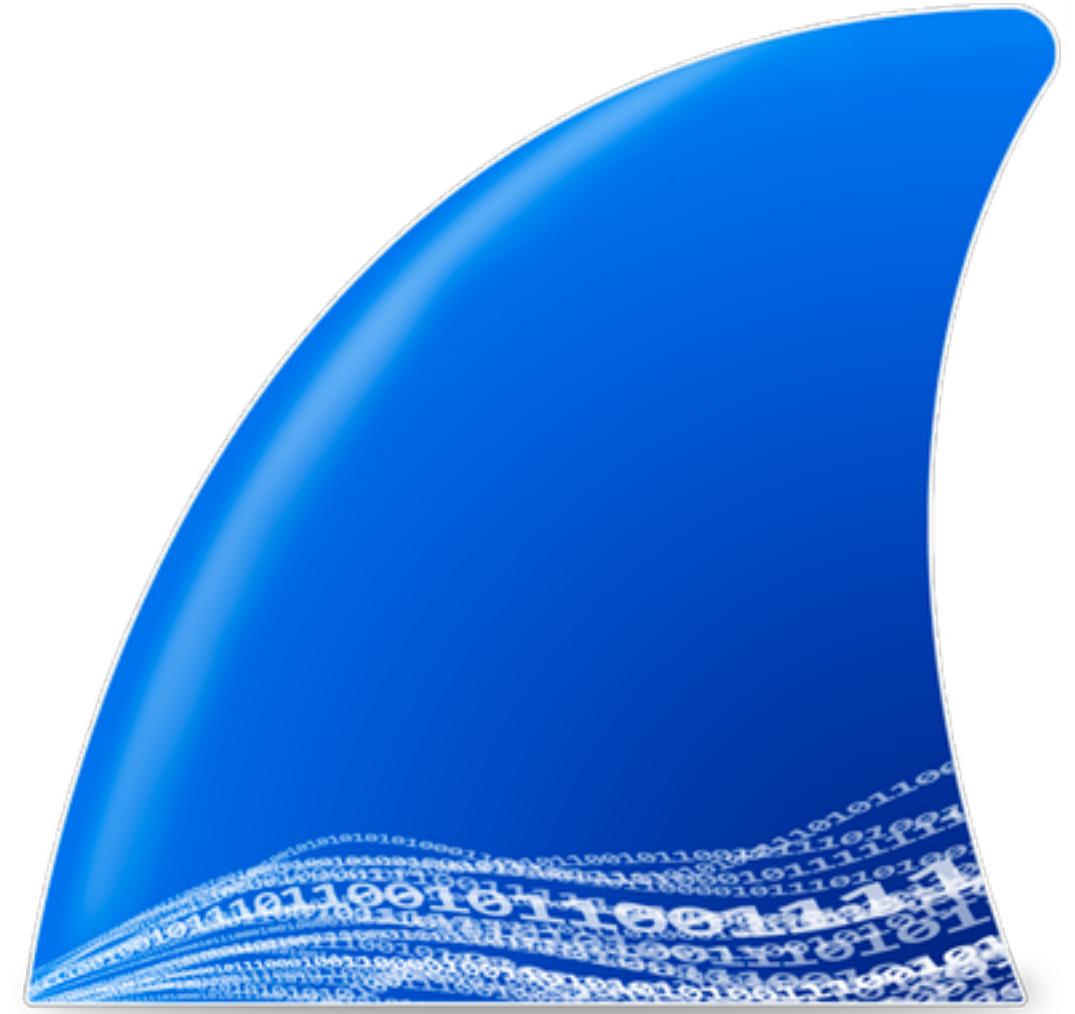
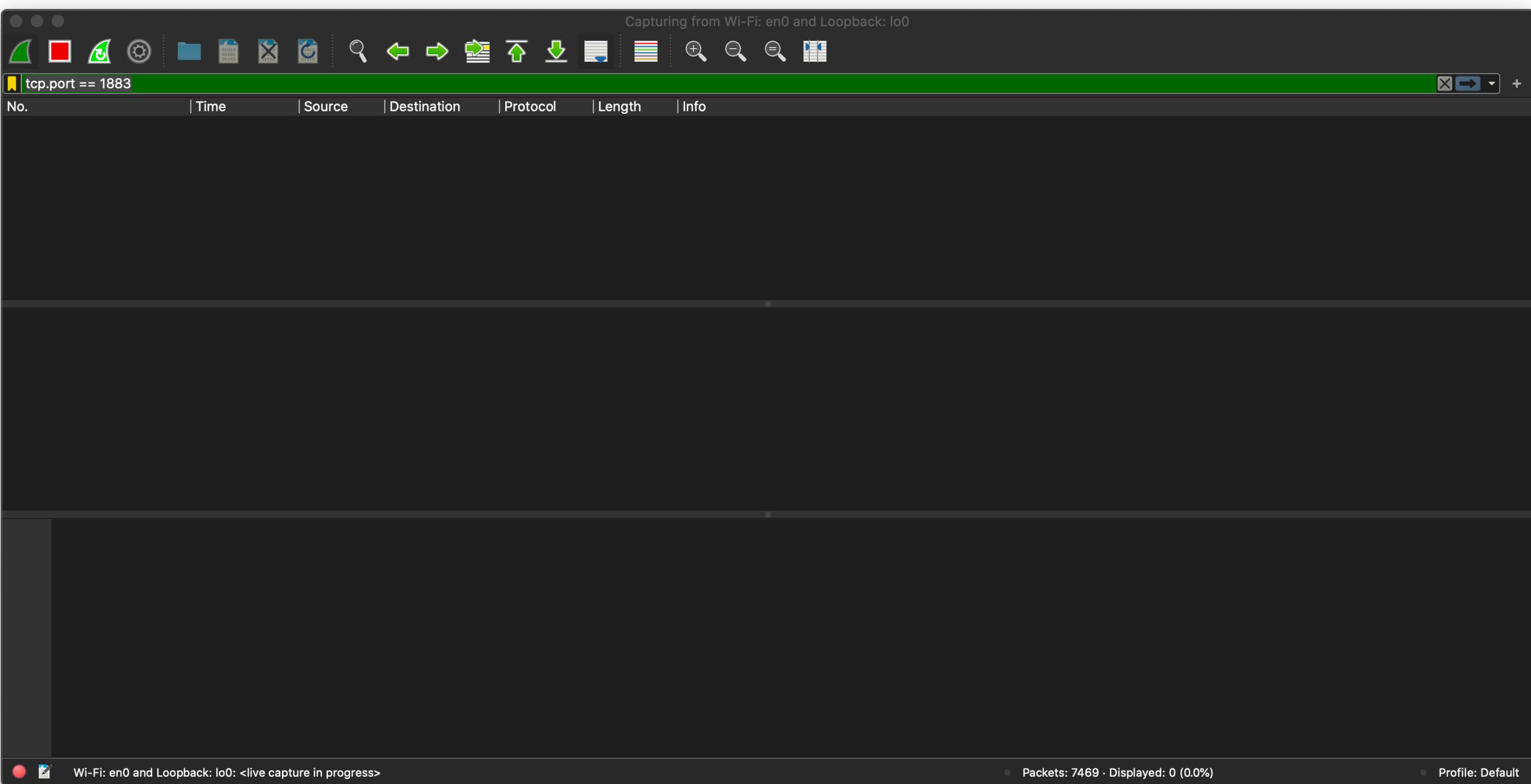
```
Opslaan voltooid.

Atmel SMART device 0x10010005 found
Device      : ATSAMD21G18A
Chip ID     : 10010005
Version     : v2.0 [Arduino:XYZ] Apr 19 2019 14:38:48
Address     : 8192
Pages       : 3968
```

- Bottom Status Bar:** Arduino NANO 33 IoT op /dev/cu.usbmodem141201

# Prepare Wireshark

1. Go to <https://www.wireshark.org>
2. Download and install Wireshark
3. Open Wireshark
4. Monitor your WiFi and local loopback



# ARP Poisoning:

Step 1: get available local network *interfaces*

Step 2: get local network interfaces *gateways*

Step 3: get *gateway information* found in step 2

Step 4: show the *connected devices* to the particular gateway

Step 5: select the *intended victim*

Step 6: poison ARP table (*pretend you are the gateway by broadcasting this*)

Step 7: do your analysis

Step 8: restore ARP tables

# Results

Filter: "tcp.port == 1883"

The screenshot shows the Wireshark interface with the following details:

- Filter Bar:** tcp.port == 1883
- Legend:** Wi-Fi: en0 and Loopback: lo0
- Toolbar:** Includes icons for file operations, search, and zoom.
- Table:** Shows captured packets with columns: No., Time, Source, Destination, Protocol, Length, and Info. Key entries include:
  - 3955 55.937407 arduino... ip4dab43a0... MQTT 86 Publish Message [factory/indicatorLightState]
  - 4009 56.984987 arduino... ip4dab43a0... TCP 86 [TCP Retransmission] 64043 → mqtt(1883) [PSH, ACK] Seq=1 Ack=1 Win=5691 Len=32
  - 4138 60.129684 arduino... ip4dab43a0... TCP 86 [TCP Retransmission] 64043 → mqtt(1883) [PSH, ACK] Seq=1 Ack=1 Win=5691 Len=32
  - 4530 65.936120 arduino... ip4dab43a0... TCP 86 [TCP Retransmission] 64043 → mqtt(1883) [PSH, ACK] Seq=1 Ack=1 Win=5691 Len=32
  - 5365 75.858333 arduino... ip4dab43a0... MQTT 634 Publish Message [factory/touchSensorState], Publish Message [factory/indicatorLightState], Publish Mes.
  - 5366 75.858336 arduino... ip4dab43a0... TCP 58 54976 → mqtt(1883) [SYN] Seq=0 Win=5744 Len=0 MSS=1436
  - 5423 76.352188 arduino... ip4dab43a0... TCP 54 64043 → mqtt(1883) [ACK] Seq=614 Ack=2 Win=5690 Len=0
  - 5437 76.505751 ip4dab43... arduino-2f6... TCP 54 mqtt(1883) → 64043 [FIN, ACK] Seq=1 Ack=1 Win=64075 Len=0
  - 5460 77.016208 ip4dab43... arduino-2f6... TCP 54 [TCP Retransmission] mqtt(1883) → 64043 [FIN, ACK] Seq=1 Ack=1 Win=64075 Len=0
  - 5632 77.955662 arduino... ip4dab43a0... TCP 86 [TCP Retransmission] 64043 → mqtt(1883) [PSH, ACK] Seq=1 Ack=2 Win=5690 Len=32
- Details Panel:** Shows expanded view of selected packet (Frame 5365), including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and MQ Telemetry Transport Protocol details.
- Hex Editor:** Shows the raw hex and ASCII data for the selected packet.
- Statistics Panel:** Shows 7589 total packets and 24 displayed.
- Bottom Status Bar:** Profile: Default

# Other types of network 'misuse'

DNS Poisoning

TCP/IP Hijacking

etc. etc.

See e.g. [https://www.tutorialspoint.com/ethical\\_hacking/ethical\\_hacking\\_dns\\_poisoning.htm](https://www.tutorialspoint.com/ethical_hacking/ethical_hacking_dns_poisoning.htm)

See also: <https://github.com/ammarx/ARP-spoofing>

# SSL and Let's Encrypt

<https://medium.com/@flynam/securing-your-iot-device-using-ssl-4643110ab901>

See also the PDF “Study of the Development of an IoT-based sensor platform for E-Agriculture”

# Links

Ethical Hacking:

[https://www.tutorialspoint.com/ethical\\_hacking/index.htm](https://www.tutorialspoint.com/ethical_hacking/index.htm)

ARP Poisoning:

[https://www.tutorialspoint.com/ethical\\_hacking/ethical\\_hacking\\_arp\\_poisoning.htm](https://www.tutorialspoint.com/ethical_hacking/ethical_hacking_arp_poisoning.htm)

Engima:

[https://en.wikipedia.org/wiki/Cryptanalysis\\_of\\_the\\_Engima](https://en.wikipedia.org/wiki/Cryptanalysis_of_the_Engima)

“

**That's all...**

**Any questions?**

**Next week: Safety...**