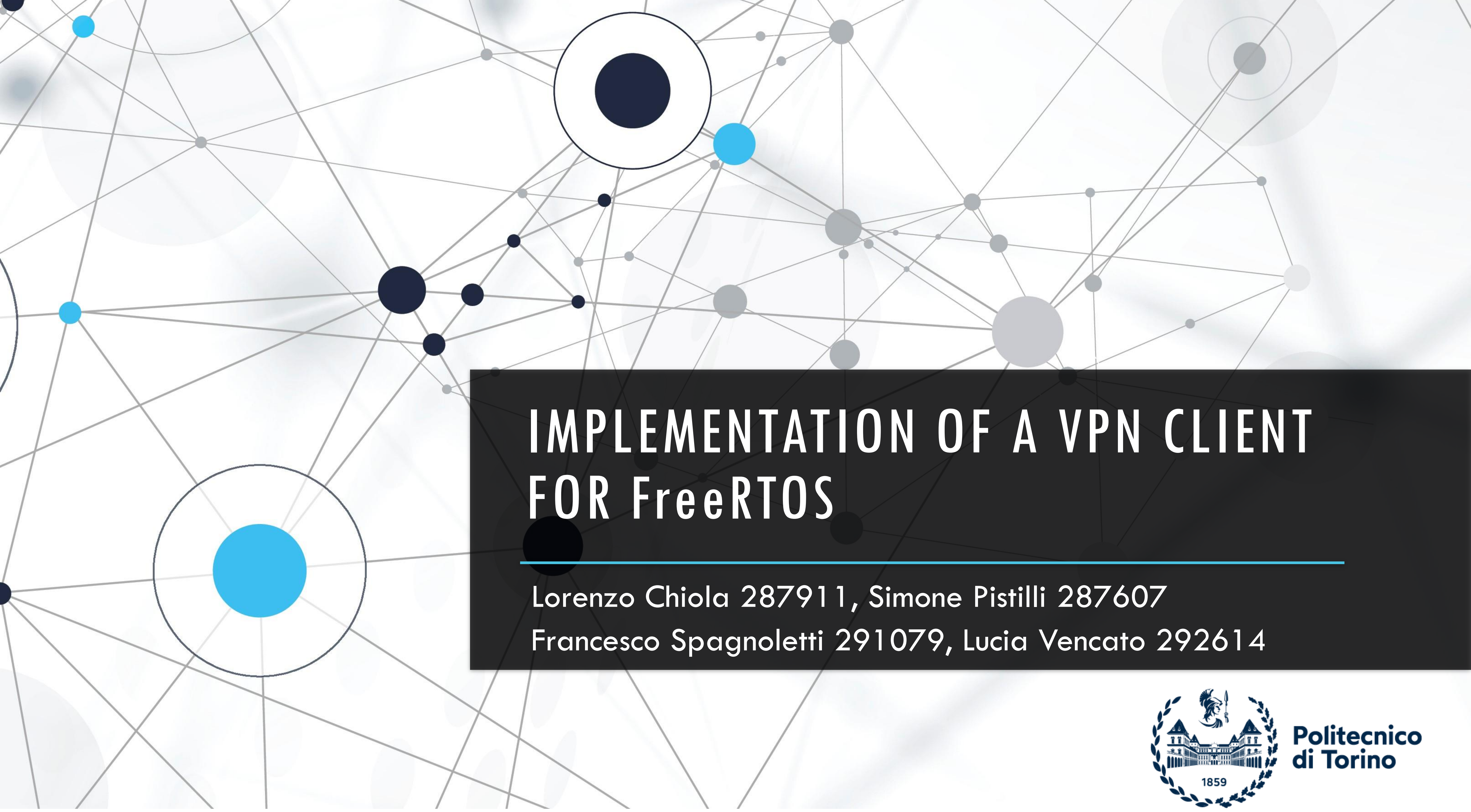


IMPLEMENTATION OF A VPN CLIENT FOR FreeRTOS

Lorenzo Chiola 287911, Simone Pistilli 287607
Francesco Spagnoletti 291079, Lucia Vencato 292614

Politecnico di Torino

OUTLINE

- VPN: WireGuard
- Project Goal & Development
- Project Demo
- Performance & Resources

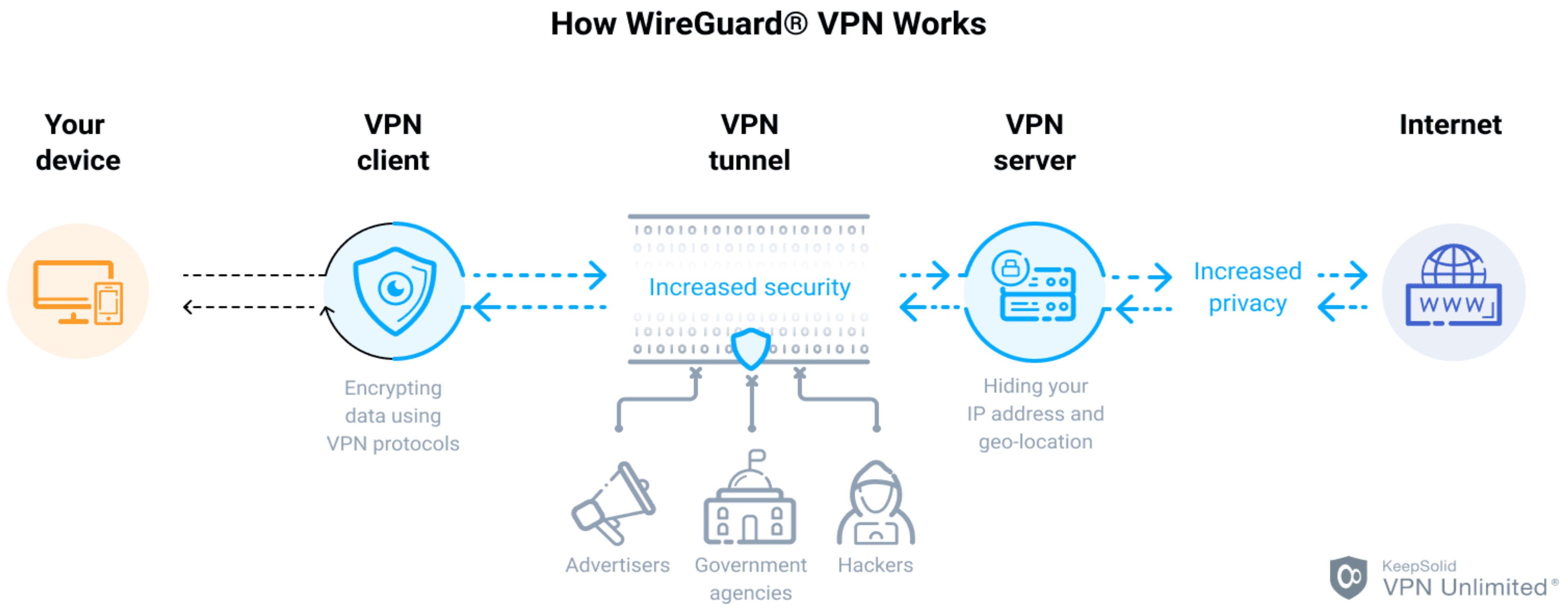
VPN

"A **virtual private network**, or VPN, is an encrypted connection over the Internet from a device to a network." [1]

A VPN is created by establishing a virtual point-to-point connection through the use of dedicated circuits or with tunneling protocols over existing networks.

[1] https://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-clients/what-is-vpn.html

WIREGUARD



[2] https://www.vpnunlimited.com/help/vpn-protocols/wireguard-protocol

WHY WIREGUARD ?

- Free and Open Source
- Extremely simple modern VPN
- State-of-the-art cryptography
- Securely encapsulates IP packets over UDP

WIREGUARD: Cryptographic Algorithms

- BLAKE2S
  - Cryptographic hash function optimized for 8 to 32-bit platforms
  - Produces digests of any size between 8 and 256 bits
- X25519
  - An elliptic curve Diffie-Hellman key exchange using Curve25519
  - It allows two parties to jointly agree on a shared secret using a non-secure channel
- CHACHA20-POLY1305
  - An Authenticated Encryption with Additional Data (AEAD) algorithm
  - ChaCha20-Poly1305 is an algorithm that combines the ChaCha20 stream chiper with the Poly1305 message authentication code

WIREGUARD: Cryptographic Algorithms

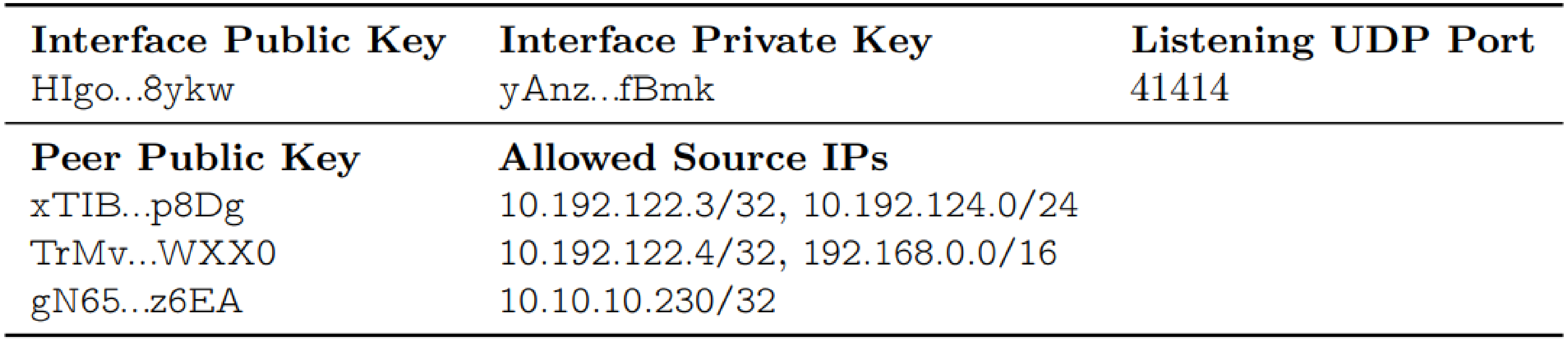
- An Authenticated Encryption with Additional Data (AEAD) algorithm guarantees both confidentiality (through encryption) as well as integrity and authenticity of data.
- Encryption only provides confidentiality but the message sent is not protected against modification.
- Additional Data must be transmitted along with the message to authenticate it. For AEAD this operation takes the form of a MAC (Message Authentication Code).
- Keyed hash functions are usually used to generate MACs.

WIREGUARD: Where to start

- Create a virtual network Interface with a specific IP Address
- Define a pair of keys for the interface
- Choose a UDP listening port
- Define all the needed Peers
  - Every Peer has a unique Public Key and for each of them a range of Allowed IPs is defined

WIREGUARD: Interface example

Cryptokey Routing Table for a WireGuard network interface



Interface Public Key	Interface Private Key	Listening UDP Port
HIgo...8ykw	yAnz...fBmk	41414

Peer Public Key	Allowed Source IPs
xTIB...p8Dg	10.192.122.3/32, 10.192.124.0/24
TrMv...WXX0	10.192.122.4/32, 192.168.0.0/16
gN65...z6EA	10.10.10.230/32

OUTLINE

- VPN: WireGuard
- Project Goal & Development
- Project Demo
- Performance & Resources

PROJECT GOAL

- Create a VPN client for FreeRTOS
  - Get a WireGuard client module
  - Port it to FreeRTOS
  - Test it on some platform
    - Simulated on Linux/Windows
    - Simulated on Qemu
    - Physical board (ESP32 was chosen)
  - Create an application that interacts with a WireGuard server

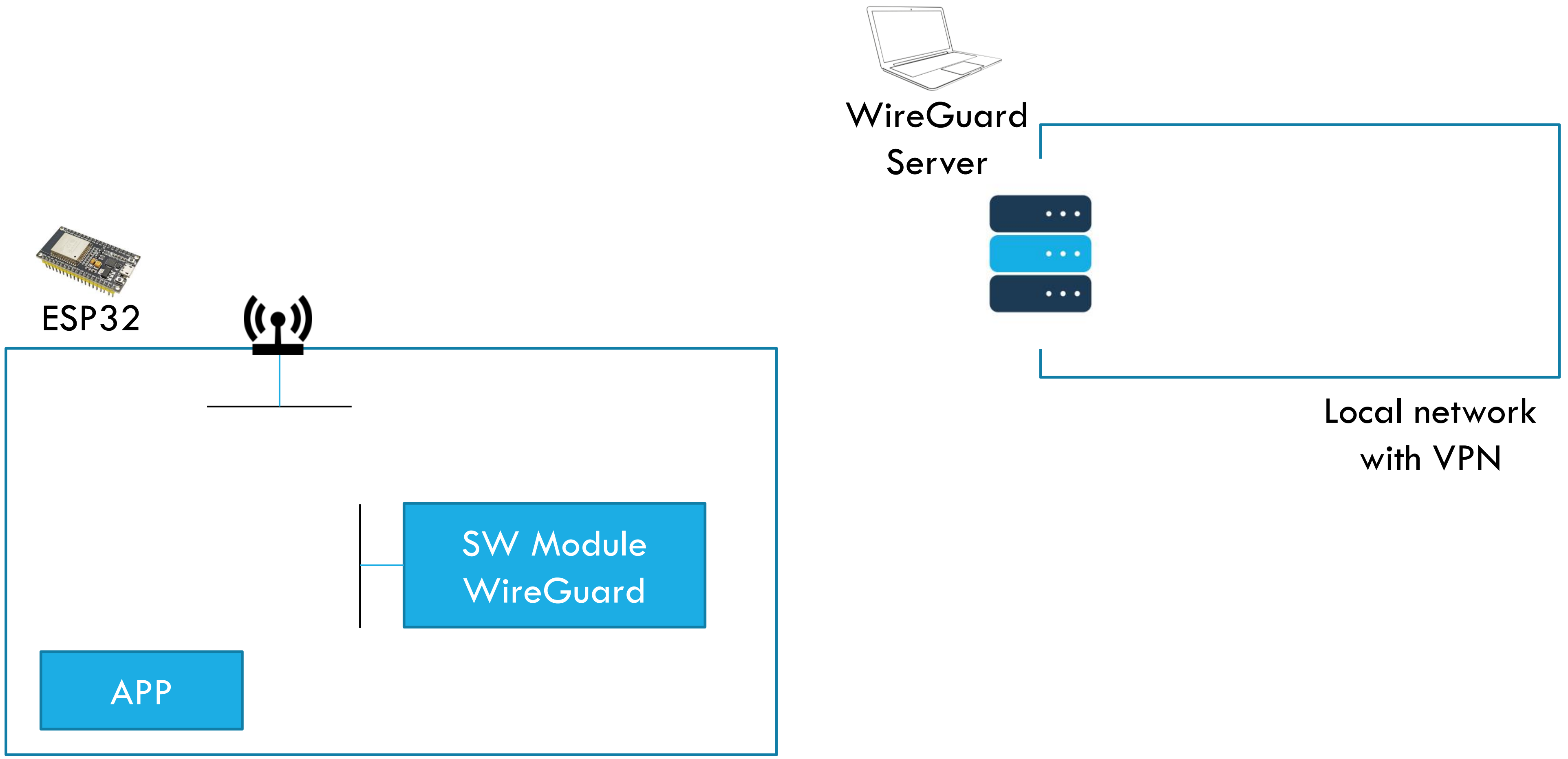
ESP32 MODULE

- Dual-core Xtensa LX6 32 bit (frequency up to 240 MHz)
- 520 KB of SRAM
- 4 MB of Flash memory
- **WiFi 802.11n**
- Bluetooth 4.2
- Peripherals: SPI, I2C, I2S, UART, CAN 2.0, Ethernet MAC, ADC 12bit
- 32 programmable GPIOs
- **Hardware True Random Number Generator**

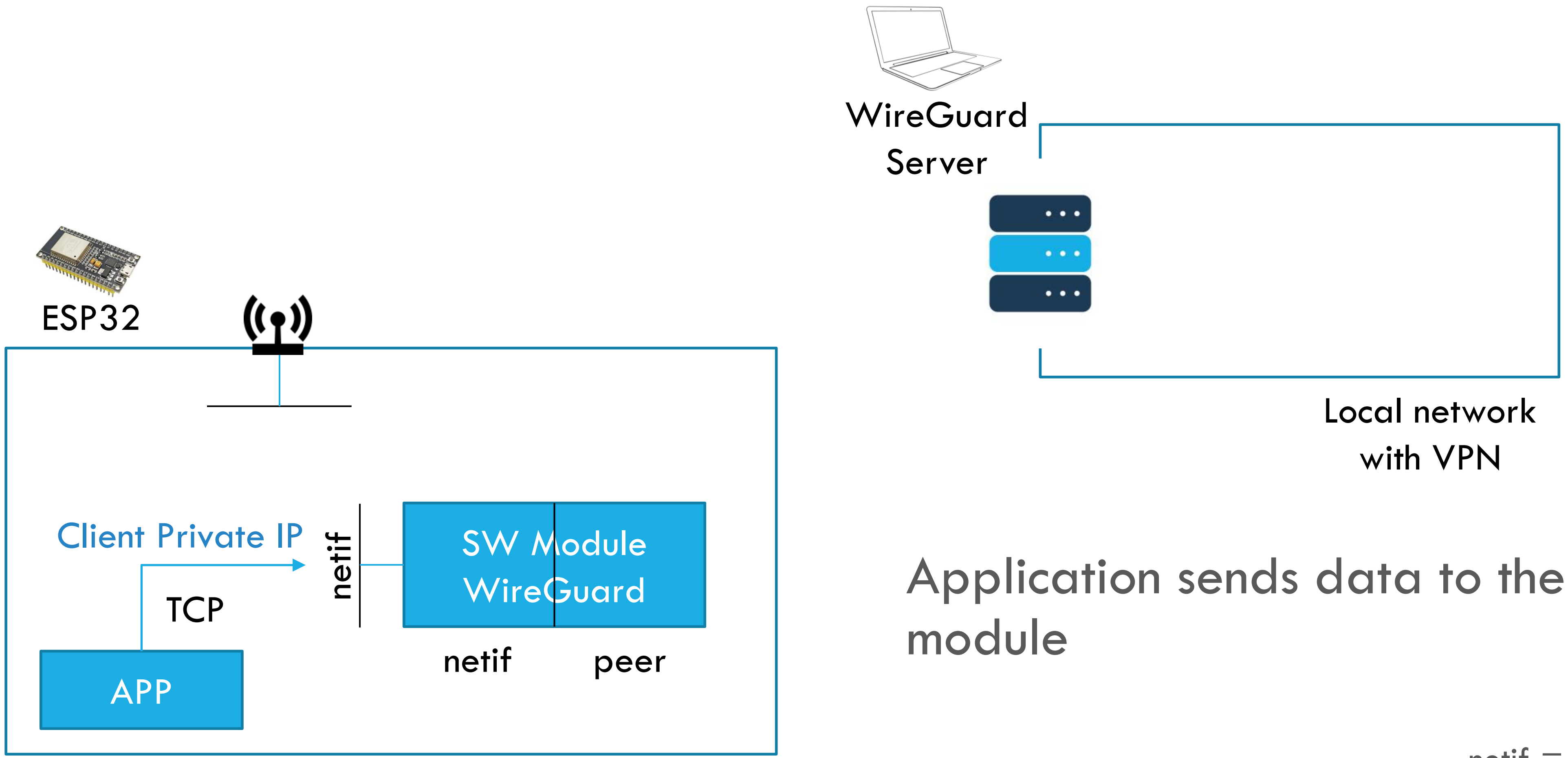
ESP32 MODULE

- ESP-IDF FreeRTOS
  - Based on Vanilla FreeRTOS v10.4.3
  - Open source lwIP lightweight IP stack
- Low price (~ 7€)

VPN WITH ESP32: project architecture



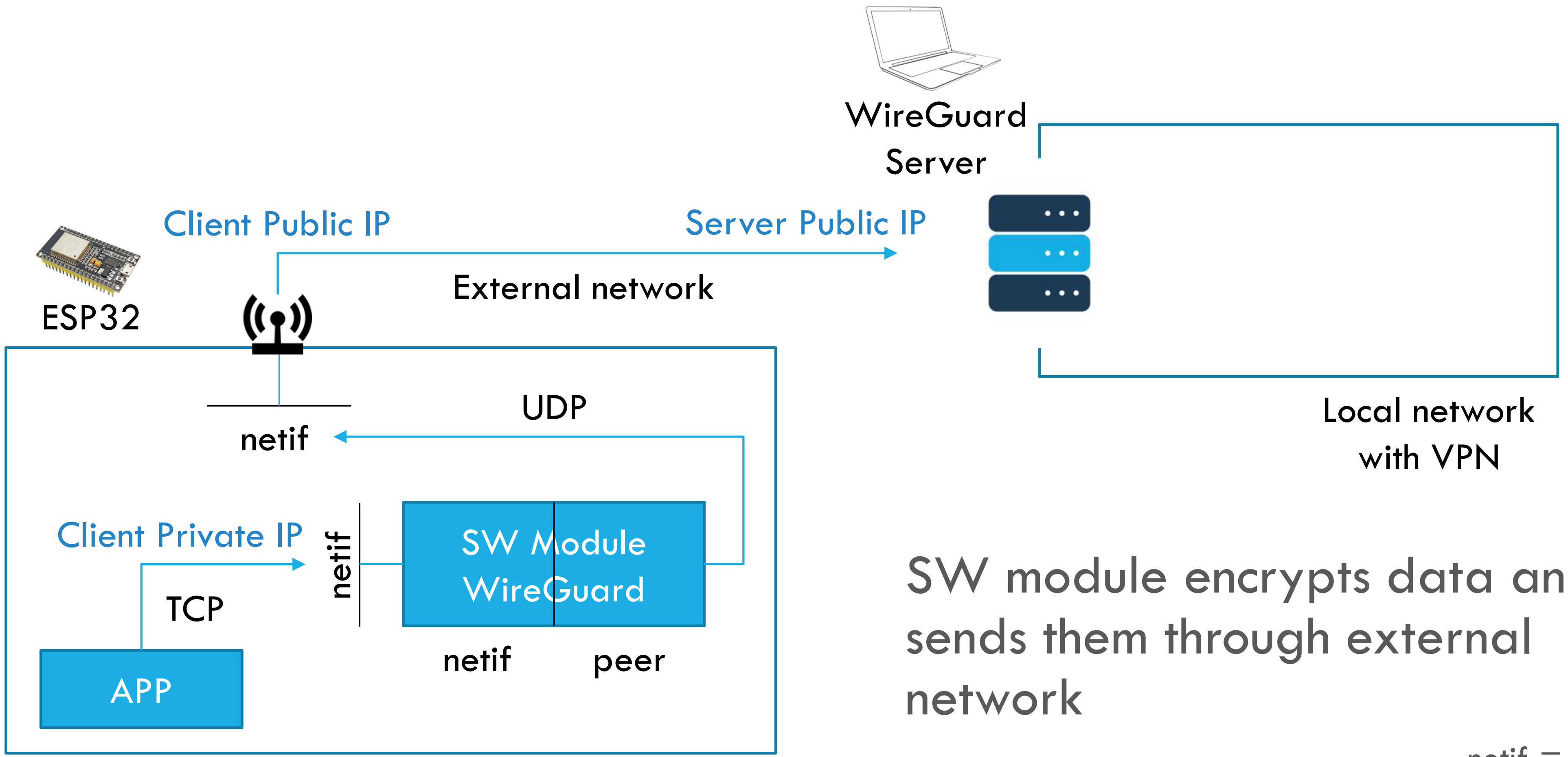
VPN WITH ESP32: project architecture



Application sends data to the SW module

netif = lwIP Network Interface

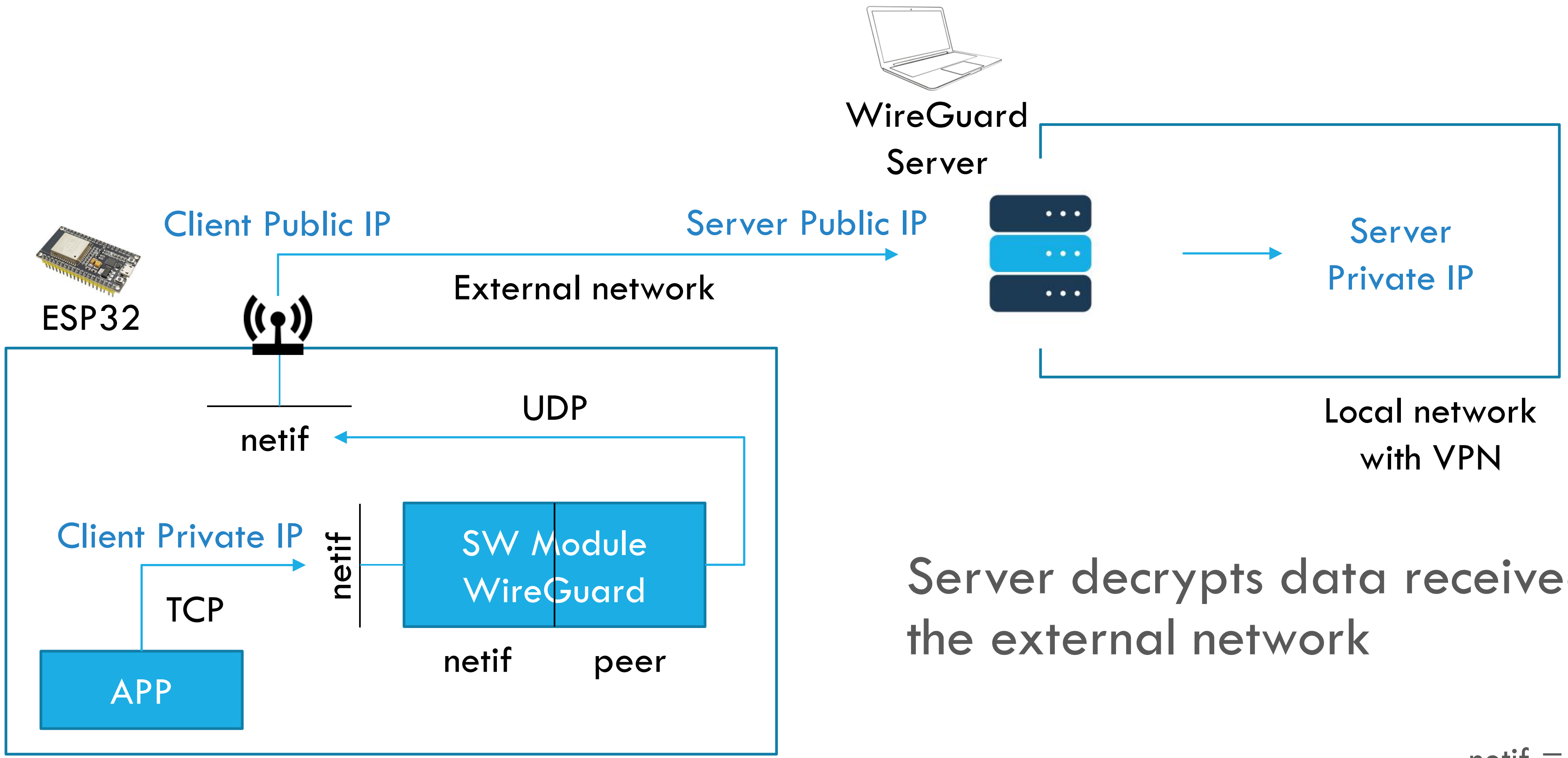
VPN WITH ESP32: project architecture



SW module encrypts data and sends them through external network

netif = lwIP Network Interface

VPN WITH ESP32: project architecture



Server decrypts data received from the external network

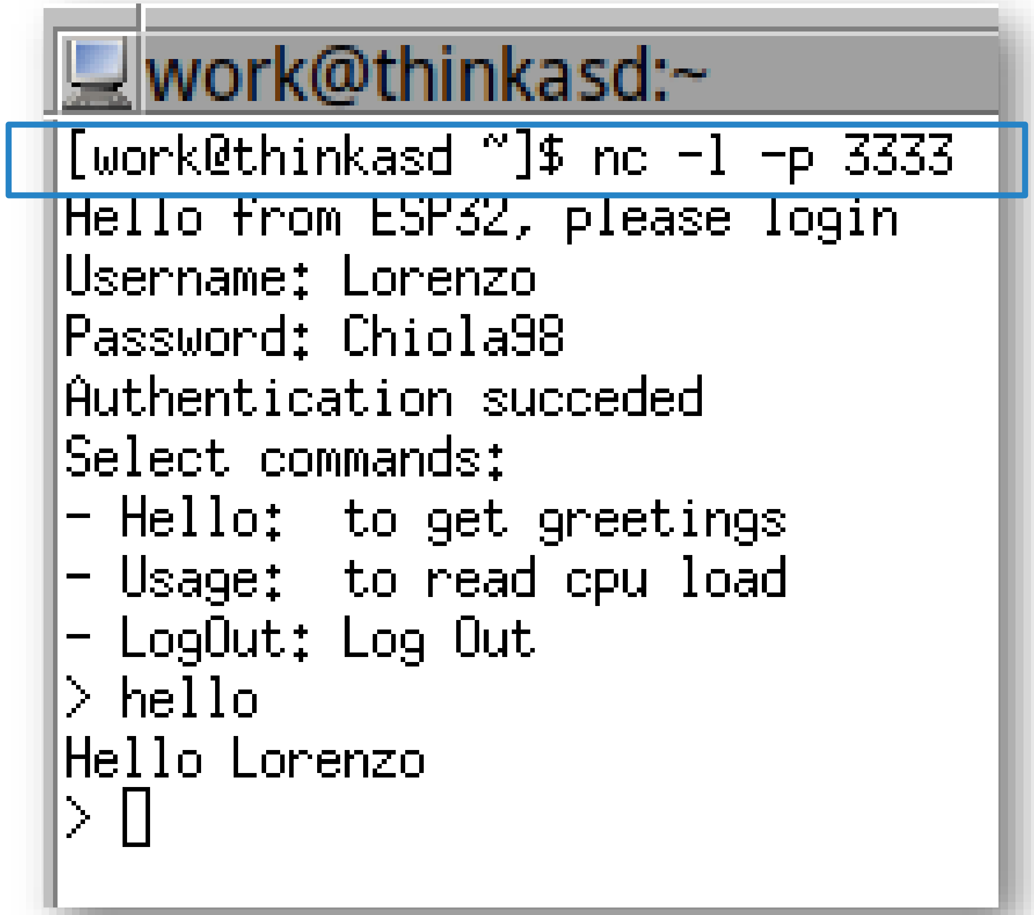
netif = lwIP Network Interface

OUTLINE

- VPN: WireGuard
- Project Goal & Development
- Project Demo
- Performance & Resources

DEMO

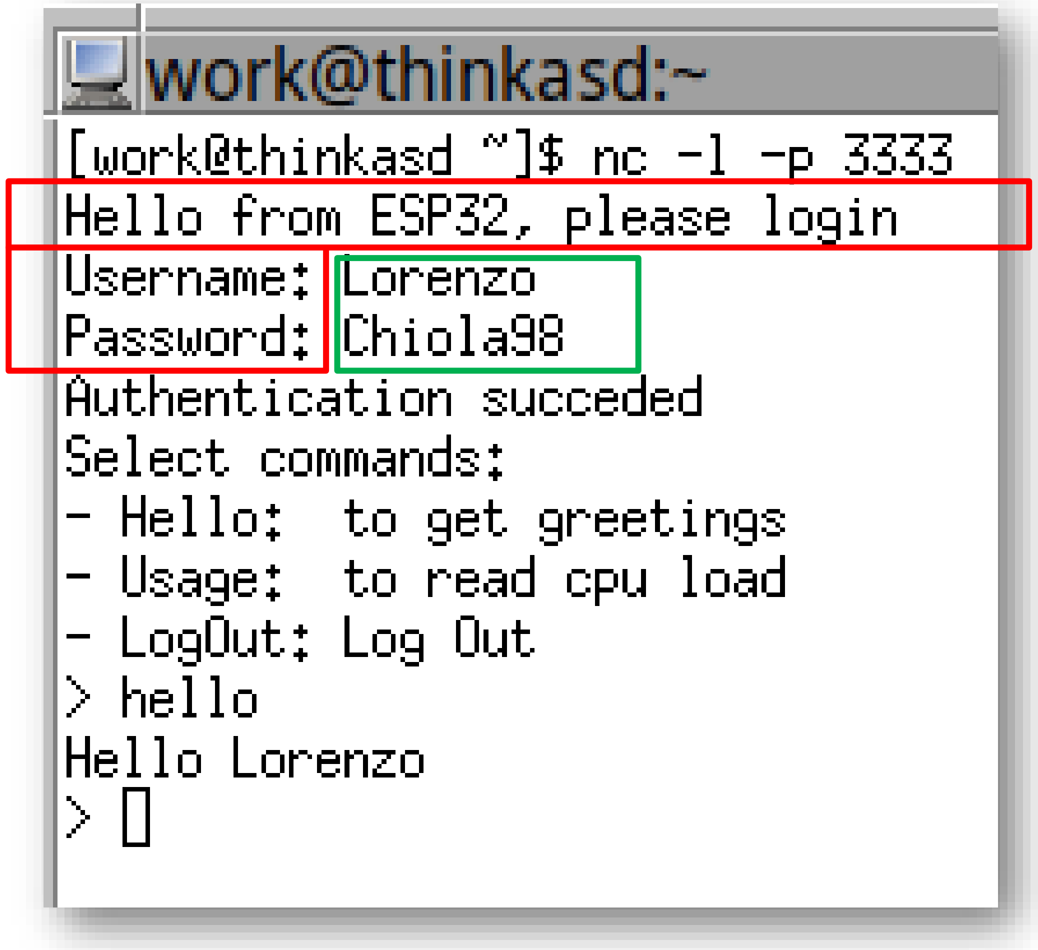
- TCP connection
  - PC configured as WireGuard Server is listening for incoming TCP packets on port 3333


work@thinkasd:~

```
[work@thinkasd ~]$ nc -l -p 3333
Hello from ESP32, please login
Username: Lorenzo
Password: Chiola98
Authentication succeded
Select commands:
- Hello:  to get greetings
- Usage:  to read cpu load
- LogOut: Log Out
> hello
Hello Lorenzo
> 
```

DEMO

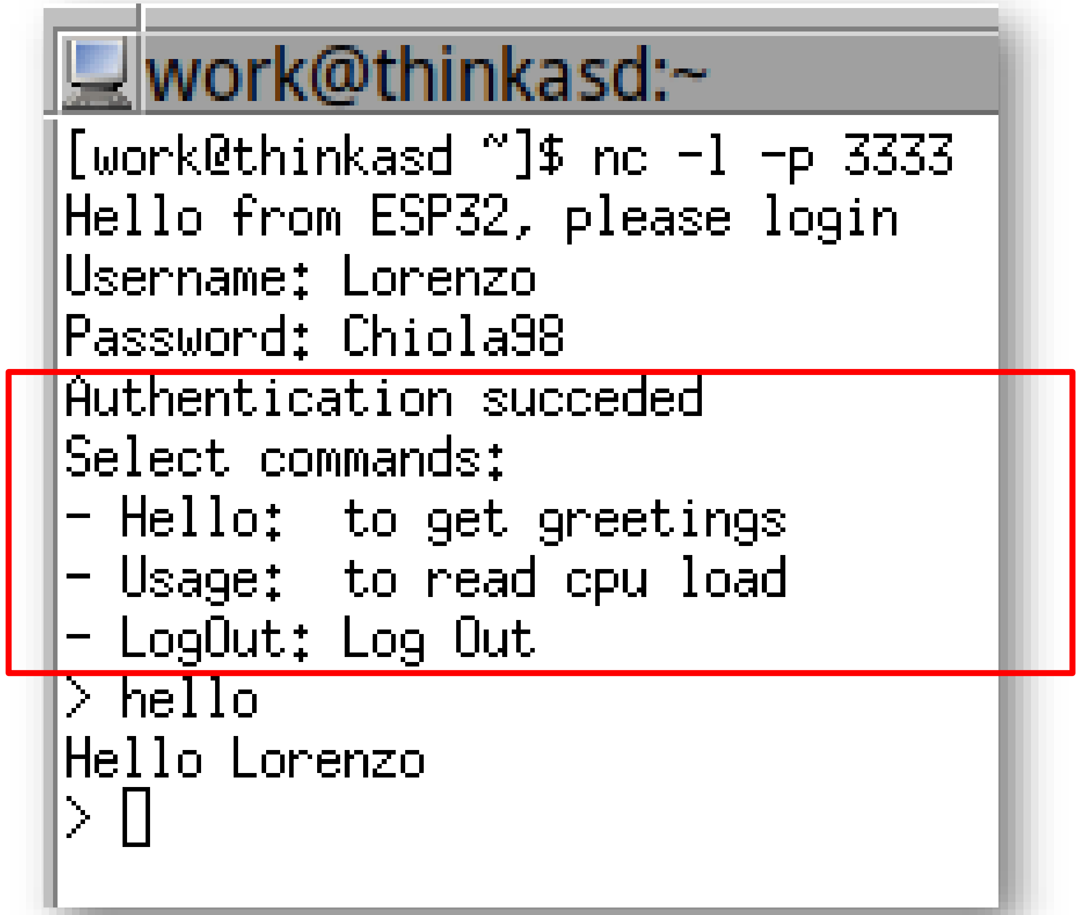
- TCP connection
  - PC configured as WireGuard Server is listening for incoming TCP packets on port 3333
- Login
  - Esp32 requests credentials (red)
  - User on PC answers (green)



```
work@thinkasd:~
[work@thinkasd ~]$ nc -l -p 3333
Hello from ESP32, please login
Username: Lorenzo
Password: Chiola98
Authentication succeded
Select commands:
- Hello:  to get greetings
- Usage:  to read cpu load
- LogOut: Log Out
> hello
Hello Lorenzo
> 
```

DEMO

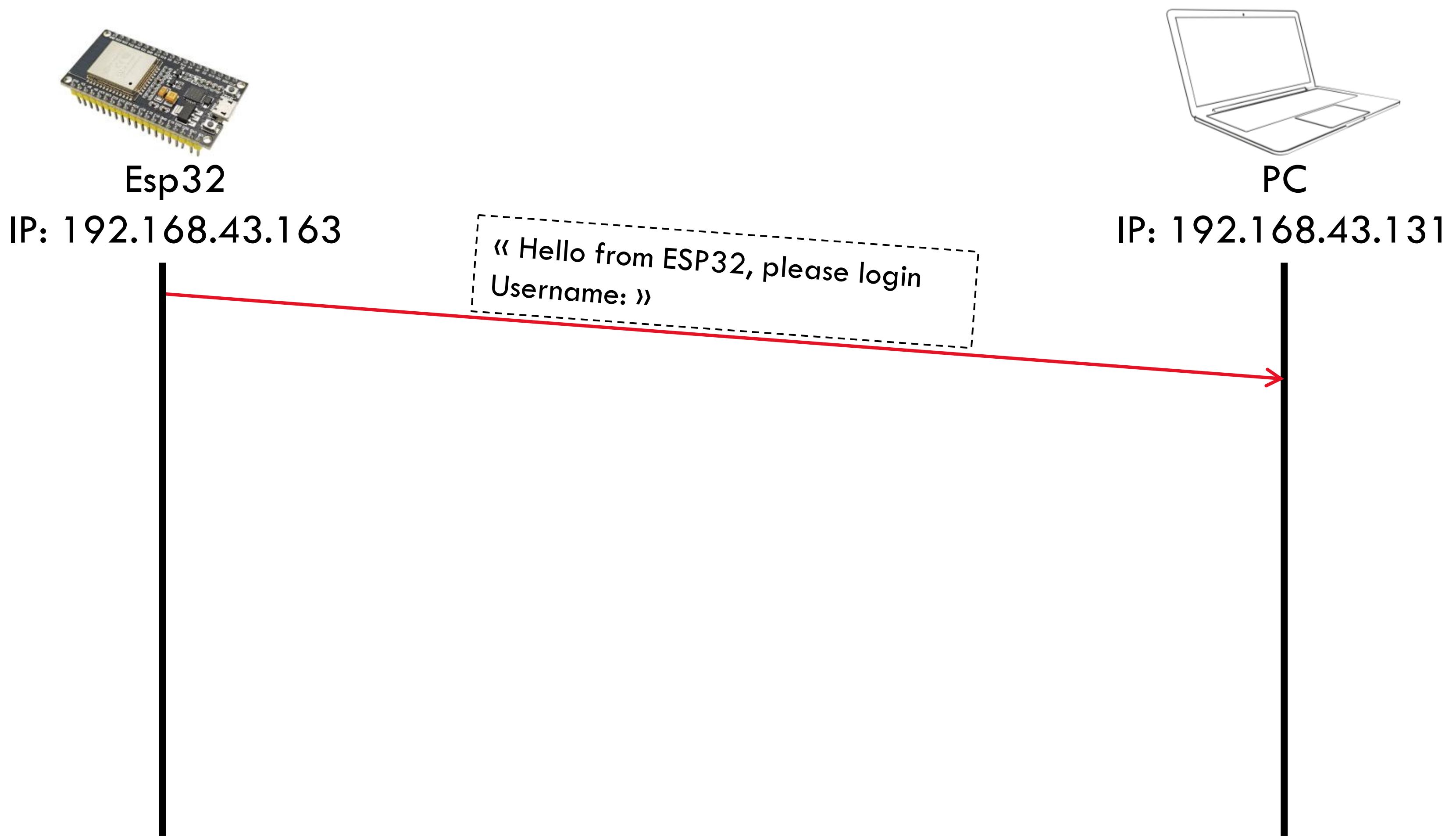
- TCP connection
  - PC configured as WireGuard Server is listening for incoming TCP packets on port 3333
- Login
  - Esp32 requests credentials
  - PC answers
- Command prompts
  - Esp32 sends available commands



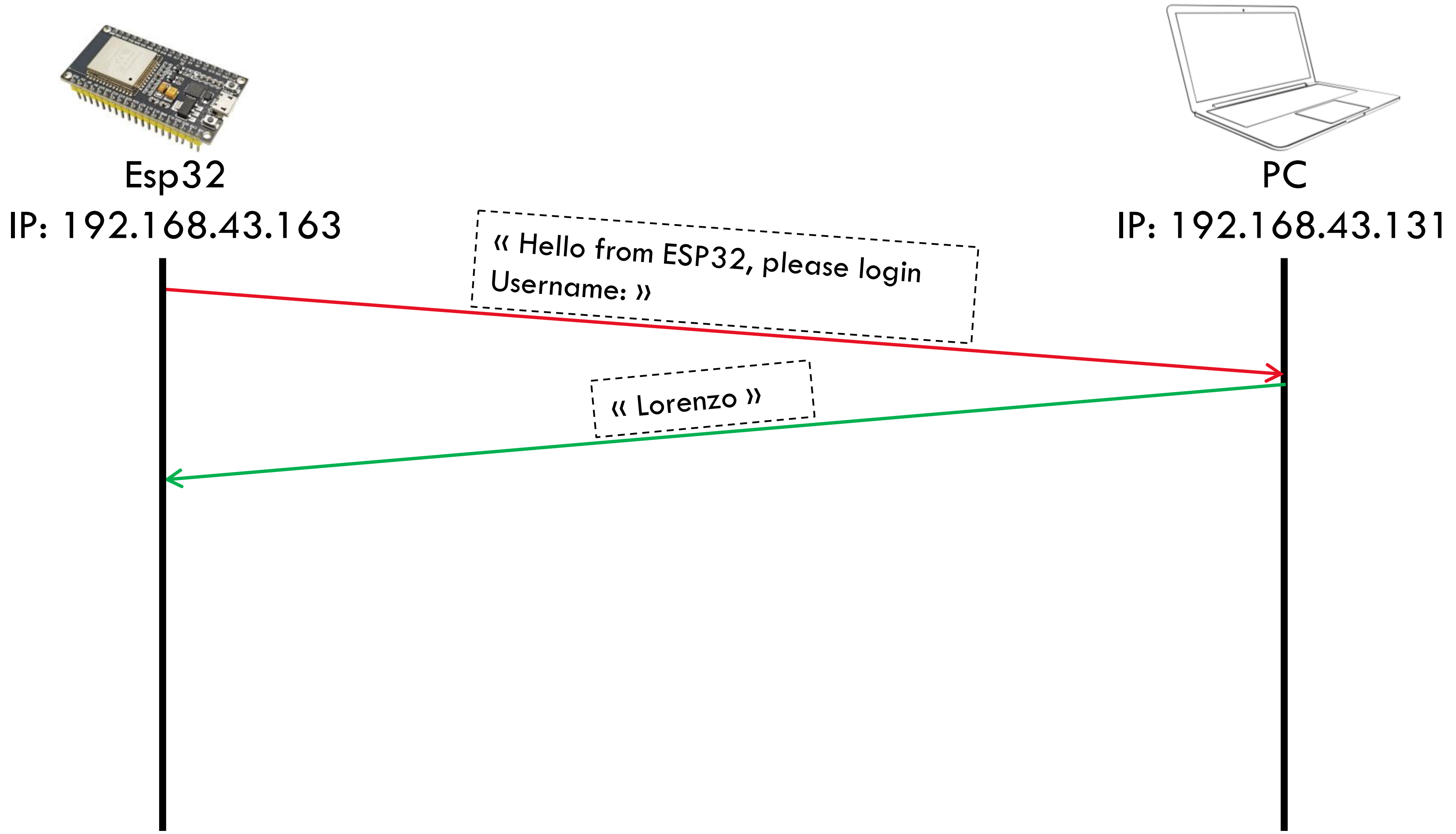
work@thinkasd:~

```
[work@thinkasd ~]$ nc -l -p 3333
Hello from ESP32, please login
Username: Lorenzo
Password: Chiola98
Authentication succeded
Select commands:
- Hello:  to get greetings
- Usage:  to read cpu load
- LogOut: Log Out
> hello
Hello Lorenzo
> 
```

DEMO: communication without WireGuard

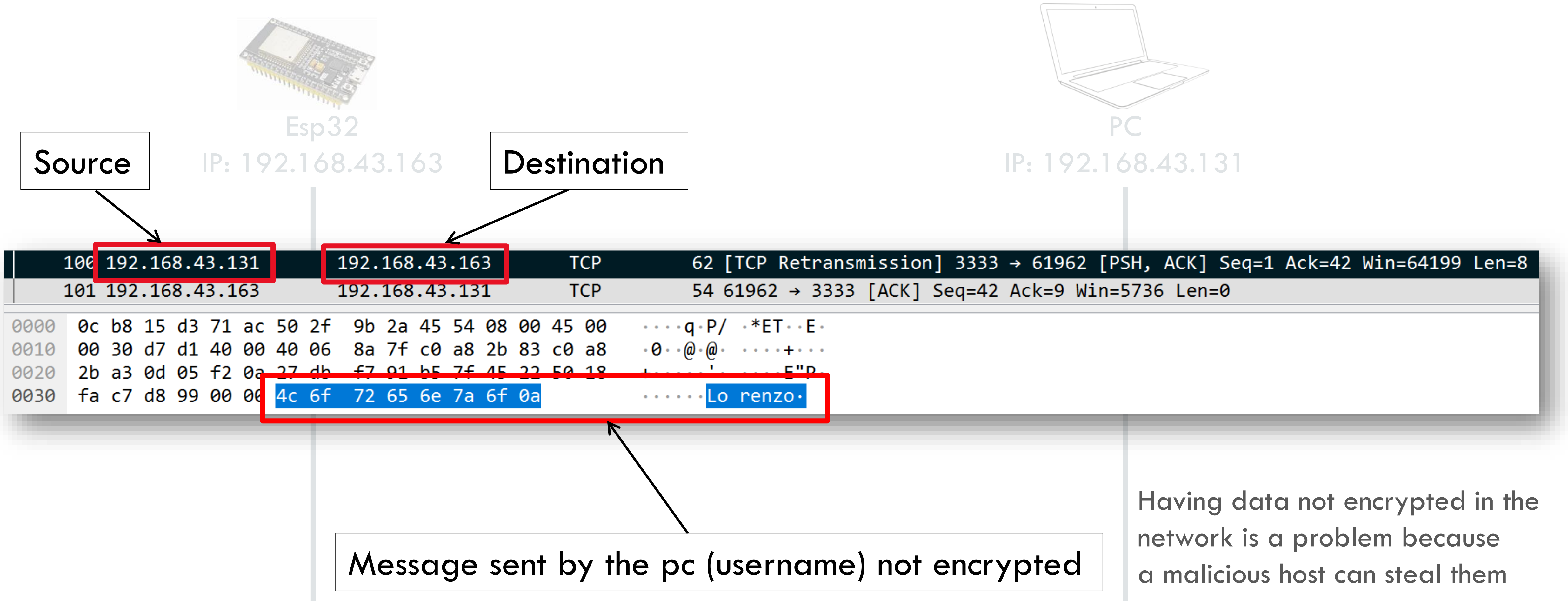


DEMO: communication without WireGuard



DEMO: communication without WireGuard

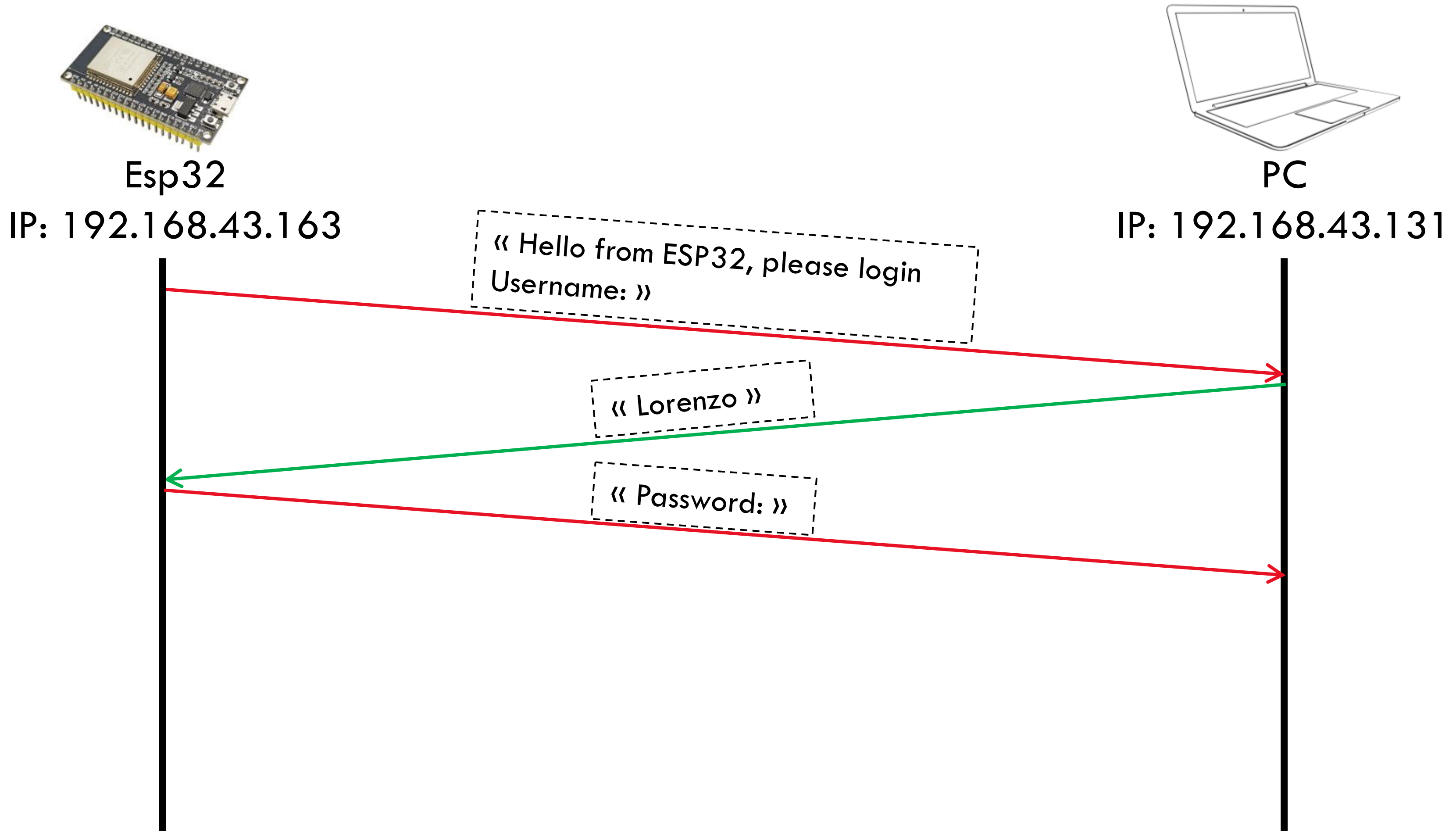
Wireshark capture



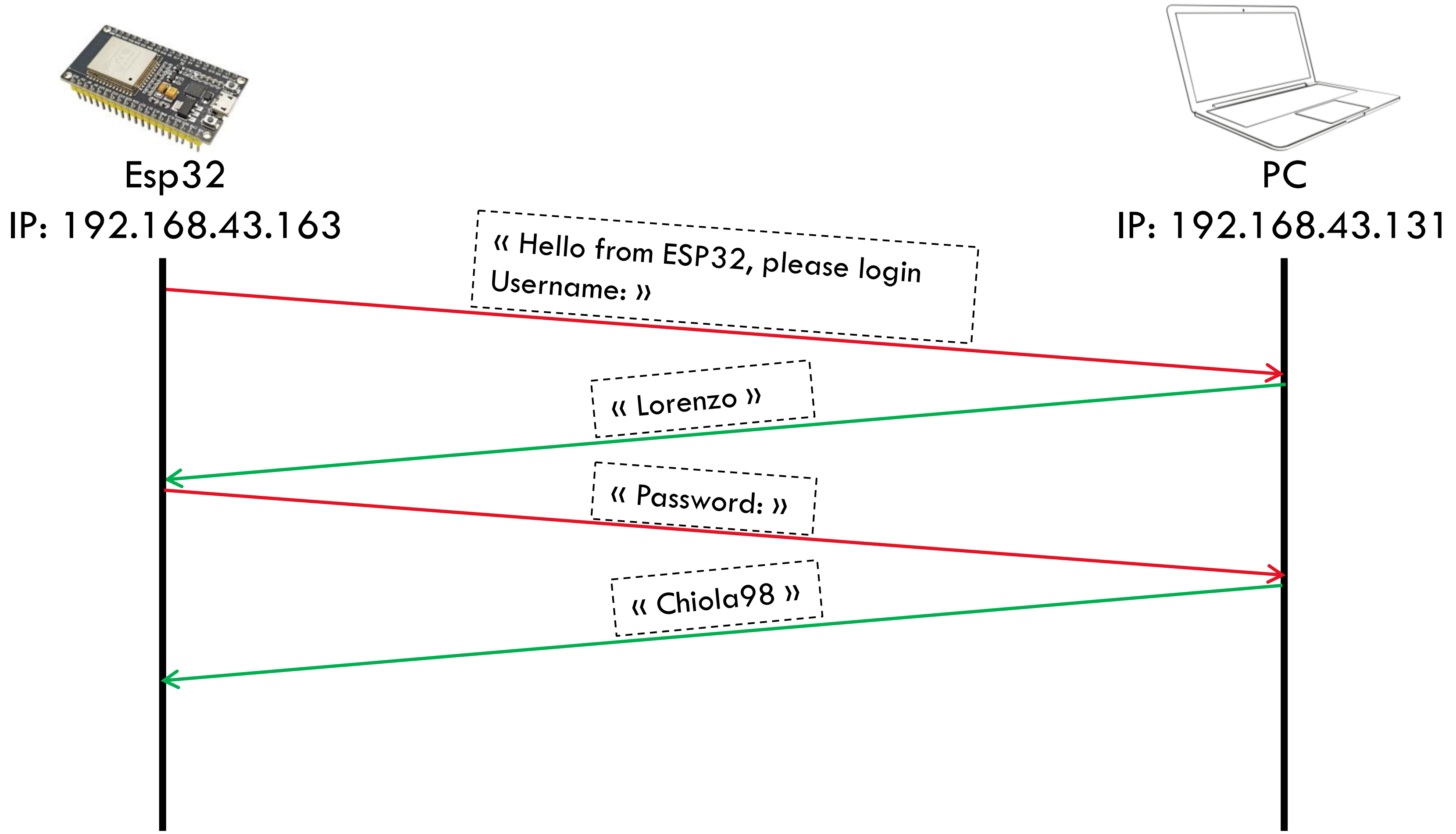
```
100 192.168.43.131     192.168.43.163    TCP       62 [TCP Retransmission] 3333 → 61962 [PSH, ACK] Seq=1 Ack=42 Win=64199 Len=8
101 192.168.43.163     192.168.43.131    TCP       54 61962 → 3333 [ACK] Seq=42 Ack=9 Win=5736 Len=0

0000  0c b8 15 d3 71 ac 50 2f  9b 2a 45 54 08 00 45 00   ····q·P/ ·*ET··E·
0010  00 30 d7 d1 40 00 40 06  8a 7f c0 a8 2b 83 c0 a8   ·0··@·@· ····+···
0020  2b a3 0d 05 f2 0a 27 db  f7 91 b5 7f 45 22 50 18   +·······'· ····E"P·
0030  fa c7 d8 99 00 00 4c 6f  72 65 6e 7a 6f 0a         ······Lo renzo·
```

Having data not encrypted in the network is a problem because a malicious host can steal them

DEMO: communication without WireGuard



DEMO: communication without WireGuard



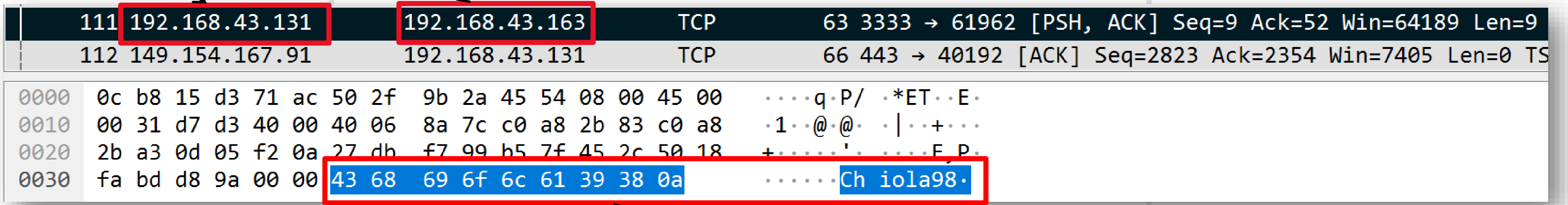
DEMO: communication without WireGuard
Wireshark capture

Esp32
IP: 192.168.43.163

PC
IP: 192.168.43.131

Source

Destination

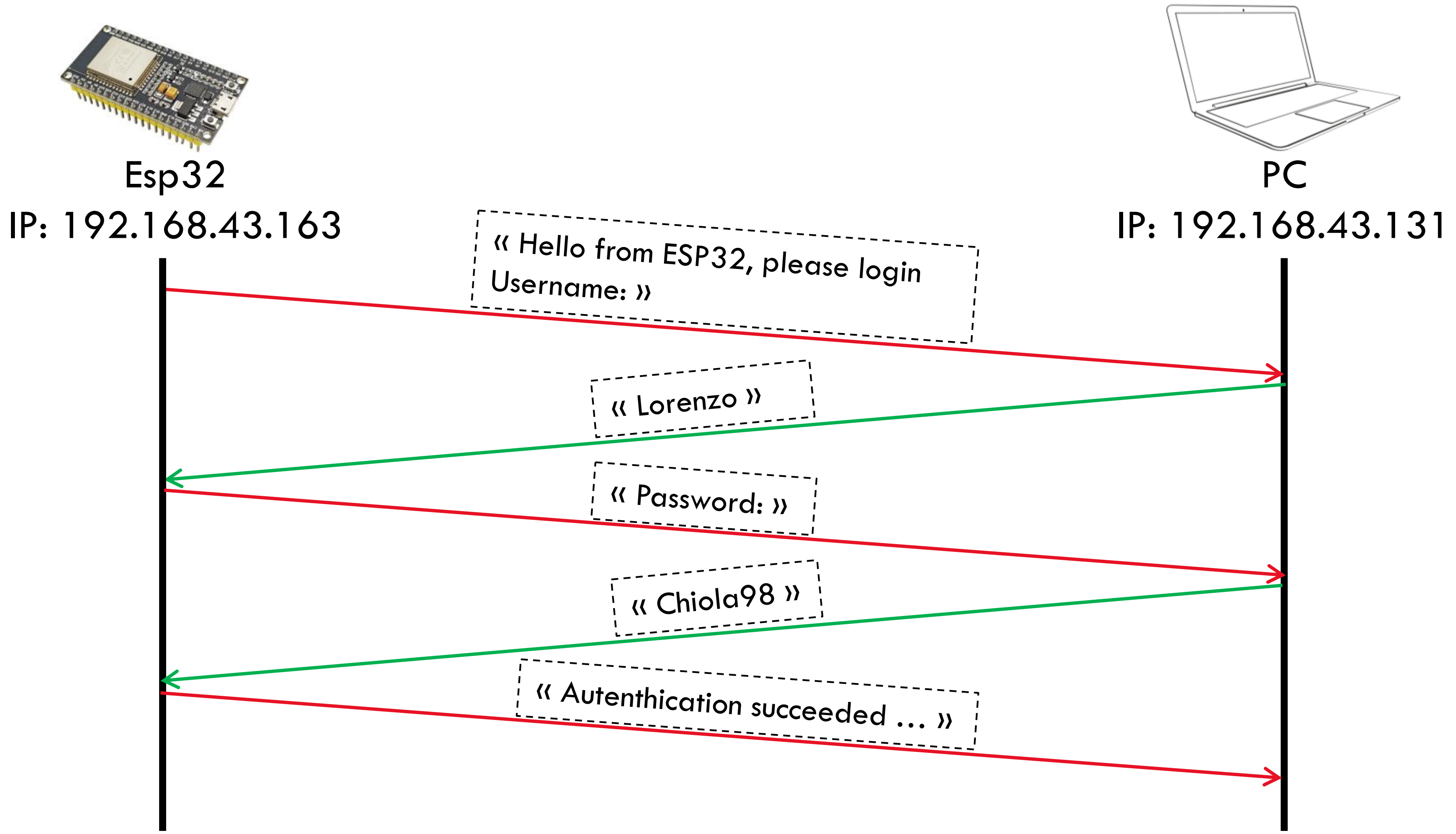


```
111 192.168.43.131    192.168.43.163    TCP    63 3333 → 61962 [PSH, ACK] Seq=9 Ack=52 Win=64189 Len=9
112 149.154.167.91    192.168.43.131    TCP    66 443 → 40192 [ACK] Seq=2823 Ack=2354 Win=7405 Len=0 TS

0000  0c b8 15 d3 71 ac 50 2f  9b 2a 45 54 08 00 45 00   ····q·P/ ·*ET··E·
0010  00 31 d7 d3 40 00 40 06  8a 7c c0 a8 2b 83 c0 a8   ·1··@·@· ·|··+···
0020  2b a3 0d 05 f2 0a 27 db  f7 99 b5 7f 45 2c 50 18   +·····'· ····E,P·
0030  fa bd d8 9a 00 00 43 68  69 6f 6c 61 39 38 0a      ······Ch iola98·
```

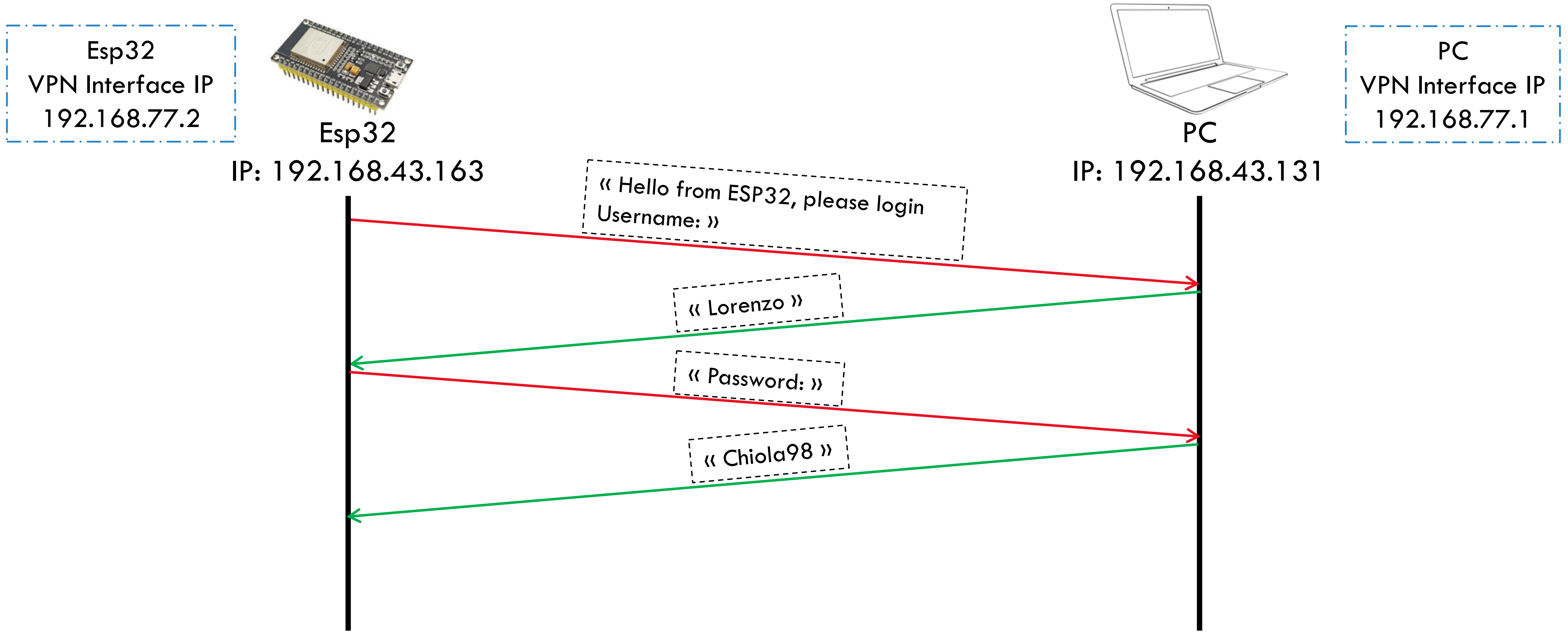
Message sent by the pc (password) not encrypted

Having data not encrypted in the network is a problem because a malicious host can steal them

DEMO: communication without WireGuard



DEMO: communication with WireGuard



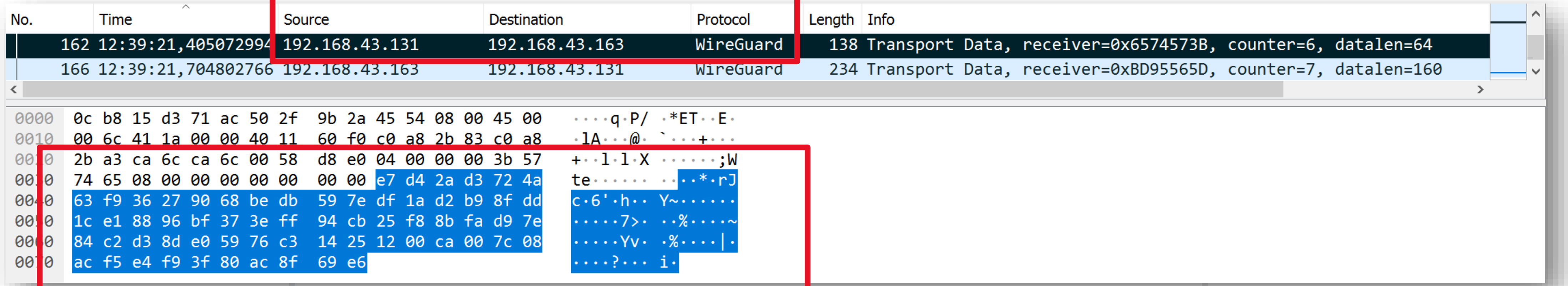
DEMO: communication with WireGuard
Wireshark capture public network

Esp32
VPN Interface IP
192.168.77.2

Esp32
IP: 192.168.43.163

PC
IP: 192.168.43.131

PC
VPN Interface IP
192.168.77.1



No.	Time	Source	Destination	Protocol	Length	Info
162	12:39:21,405072994	192.168.43.131	192.168.43.163	WireGuard	138	Transport Data, receiver=0x6574573B, counter=6, datalen=64
166	12:39:21,704802766	192.168.43.163	192.168.43.131	WireGuard	234	Transport Data, receiver=0xBD95565D, counter=7, datalen=160

```
0000  0c b8 15 d3 71 ac 50 2f  9b 2a 45 54 08 00 45 00   ····q·P/ ·*ET··E·
0010  00 6c 41 1a 00 00 40 11  60 f0 c0 a8 2b 83 c0 a8   ·lA···@· `···+···
0020  2b a3 ca 6c ca 6c 00 58  d8 e0 04 00 00 00 3b 57   +··l·l·X ······;W
0030  74 65 08 00 00 00 00 00  00 00 e7 d4 2a d3 72 4a   te······ ····*·rJ
0040  63 f9 36 27 90 68 be db  59 7e df 1a d2 b9 8f dd   c·6'·h·· Y~······
0050  1c e1 88 96 bf 37 3e ff  94 cb 25 f8 8b fa d9 7e   ·····7>· ··%····~
0060  84 c2 d3 8d e0 59 76 c3  14 25 12 00 ca 00 7c 08   ·····Yv· ·%····|·
0070  ac f5 e4 f9 3f 80 ac 8f  69 e6                     ····?··· i·
```

Message sent by the PC (password) in the local network is encrypted.

Communication is encrypted so it ensures a more secure communication

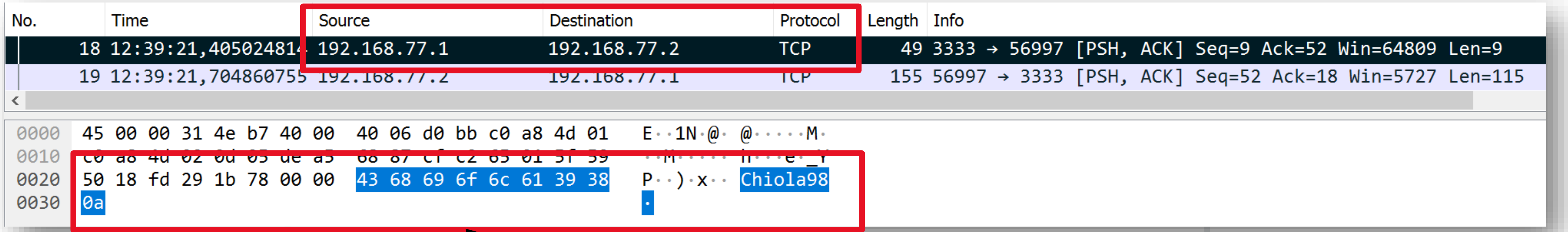
DEMO: communication with WireGuard
Wireshark capture WireGuard network interface

Esp32
VPN Interface IP
192.168.77.2

Esp32
IP: 192.168.43.163

PC
IP: 192.168.43.131

PC
VPN Interface IP
192.168.77.1

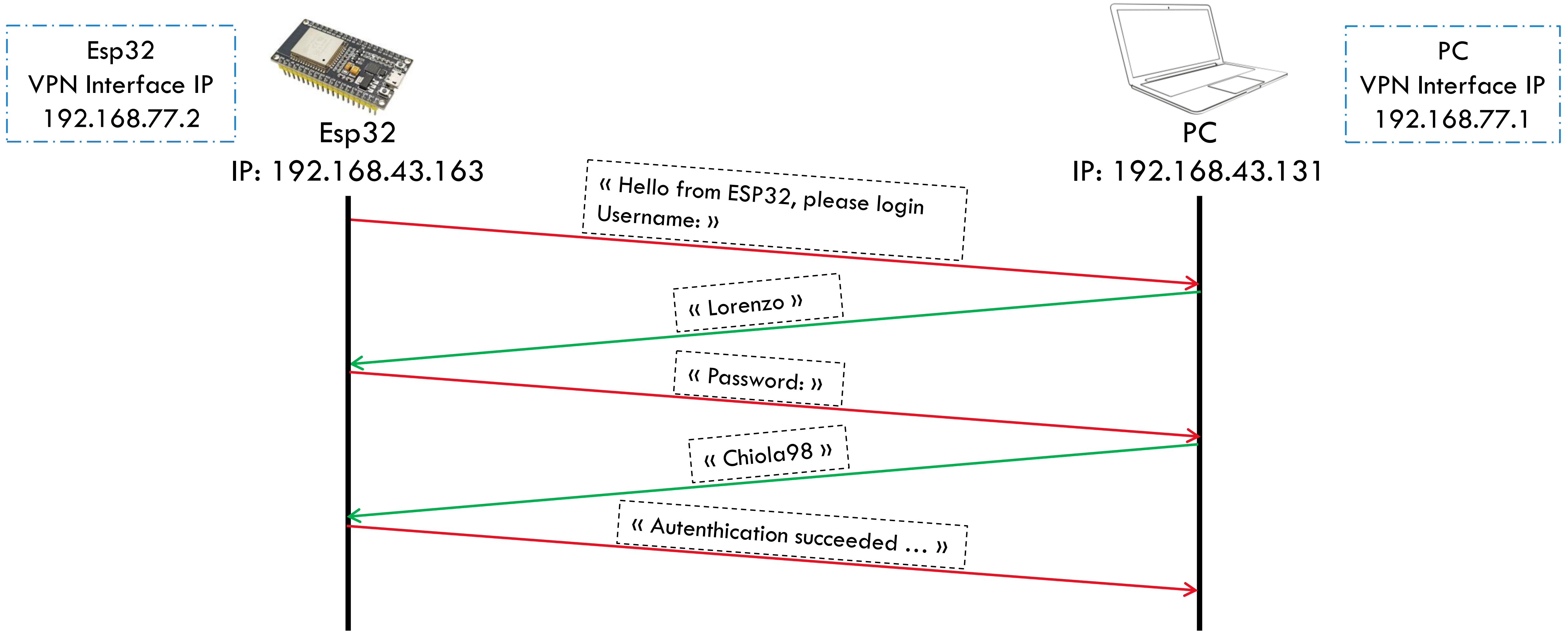


```
No.   Time                  Source          Destination     Protocol  Length  Info
   18 12:39:21,405024814    192.168.77.1    192.168.77.2    TCP           49  3333 → 56997 [PSH, ACK] Seq=9 Ack=52 Win=64809 Len=9
   19 12:39:21,704860755    192.168.77.2    192.168.77.1    TCP          155  56997 → 3333 [PSH, ACK] Seq=52 Ack=18 Win=5727 Len=115

0000  45 00 00 31 4e b7 40 00  40 06 d0 bb c0 a8 4d 01   E··1N·@· @·····M·
0010  c0 a8 4d 02 0d 05 de a5  68 87 cf c2 65 01 5f 59   ··M····· h···e·_Y
0020  50 18 fd 29 1b 78 00 00  43 68 69 6f 6c 61 39 38   P··)·x·· Chiola98
0030  0a                                                 ·
```

Message sent by the PC (password) can only be read inside the VPN tunnel.

Traffic inside the WireGuard network interface can only be seen in the PC of the user, so it's not a problem if data are readable

DEMO: communication with WireGuard



OUTLINE

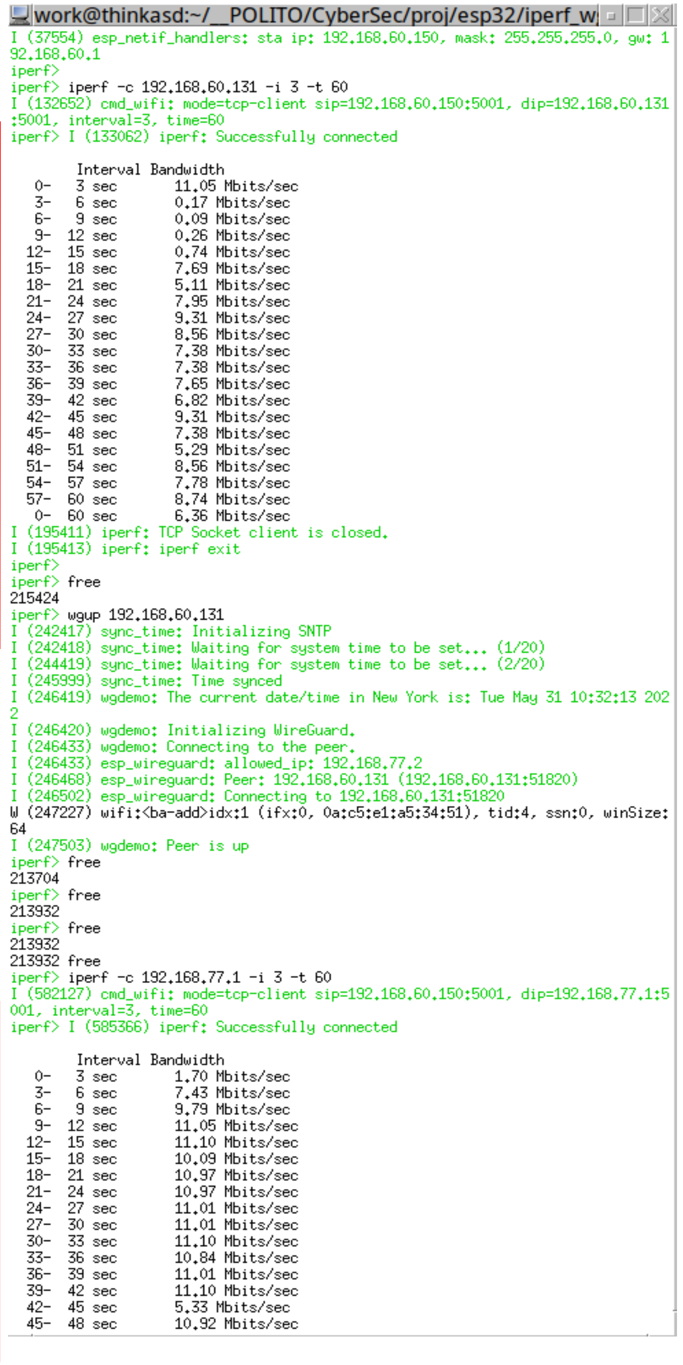
- VPN: WireGuard
- Project Goal & Development
- Project Demo
- Performance & Resources

PERFORMANCE

- **iperf**
  - Multi-platform network **throughput**
  - Measurement tool TCP (by default) connection between 2 hosts

PERFORMANCE

- PC : `iperf -s`
- Esp32 : `iperf -c <ip address> -t 60`
- Exceeds PHY speed
  - ~10 Mbit/s outside VPN
  - ~10 Mbit/s inside VPN

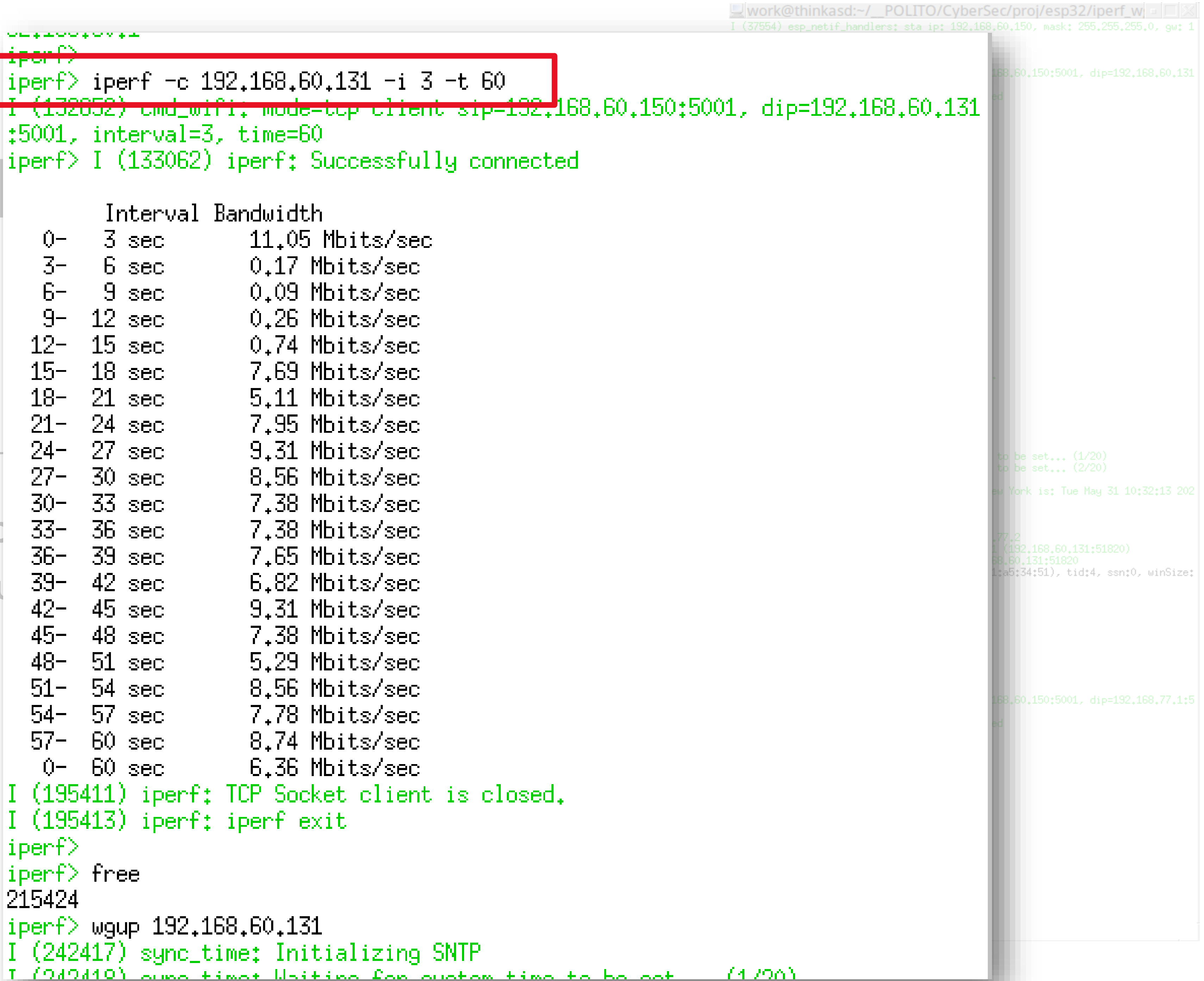


```
I (37554) esp_netif_handlers: sta ip: 192.168.60.150, mask: 255.255.255.0, gw: 1
92.168.60.1
iperf>
iperf> iperf -c 192.168.60.131 -i 3 -t 60
I (132652) cmd_wifi: mode=tcp-client sip=192.168.60.150:5001, dip=192.168.60.131
:5001, interval=3, time=60
iperf> I (133062) iperf: Successfully connected

        Interval Bandwidth
   0-   3 sec       11.05 Mbits/sec
   3-   6 sec       0.17 Mbits/sec
   6-   9 sec       0.09 Mbits/sec
   9-  12 sec       0.26 Mbits/sec
  12-  15 sec       0.74 Mbits/sec
  15-  18 sec       7.69 Mbits/sec
  18-  21 sec       5.11 Mbits/sec
  21-  24 sec       7.95 Mbits/sec
  24-  27 sec       9.31 Mbits/sec
  27-  30 sec       8.56 Mbits/sec
  30-  33 sec       7.38 Mbits/sec
  33-  36 sec       7.38 Mbits/sec
  36-  39 sec       7.65 Mbits/sec
  39-  42 sec       6.82 Mbits/sec
  42-  45 sec       9.31 Mbits/sec
  45-  48 sec       7.38 Mbits/sec
  48-  51 sec       5.29 Mbits/sec
  51-  54 sec       8.56 Mbits/sec
  54-  57 sec       7.78 Mbits/sec
  57-  60 sec       8.74 Mbits/sec
   0-  60 sec       6.36 Mbits/sec
I (195411) iperf: TCP Socket client is closed.
I (195413) iperf: iperf exit
iperf>
iperf> free
215424
iperf> wgup 192.168.60.131
I (242417) sync_time: Initializing SNTP
I (242418) sync_time: Waiting for system time to be set... (1/20)
I (244419) sync_time: Waiting for system time to be set... (2/20)
I (245999) sync_time: Time synced
I (246419) wgdemo: The current date/time in New York is: Tue May 31 10:32:13 202
2
I (246420) wgdemo: Initializing WireGuard.
I (246433) wgdemo: Connecting to the peer.
I (246433) esp_wireguard: allowed_ip: 192.168.77.2
I (246468) esp_wireguard: Peer: 192.168.60.131 (192.168.60.131:51820)
I (246502) esp_wireguard: Connecting to 192.168.60.131:51820
W (247227) wifi:<ba-add>idx:1 (ifx:0, 0a:c5:e1:a5:34:51), tid:4, ssn:0, winSize:
64
I (247503) wgdemo: Peer is up
iperf> free
213704
iperf> free
213932
iperf> free
213932
213932 free
iperf> iperf -c 192.168.77.1 -i 3 -t 60
I (582127) cmd_wifi: mode=tcp-client sip=192.168.60.150:5001, dip=192.168.77.1:5
001, interval=3, time=60
iperf> I (585366) iperf: Successfully connected

        Interval Bandwidth
   0-   3 sec       1.70 Mbits/sec
   3-   6 sec       7.43 Mbits/sec
   6-   9 sec       9.79 Mbits/sec
   9-  12 sec       11.05 Mbits/sec
  12-  15 sec       11.10 Mbits/sec
  15-  18 sec       10.09 Mbits/sec
  18-  21 sec       10.97 Mbits/sec
  21-  24 sec       10.97 Mbits/sec
  24-  27 sec       11.01 Mbits/sec
  27-  30 sec       11.01 Mbits/sec
  30-  33 sec       11.10 Mbits/sec
  33-  36 sec       10.84 Mbits/sec
  36-  39 sec       11.01 Mbits/sec
  39-  42 sec       11.10 Mbits/sec
  42-  45 sec       5.33 Mbits/sec
  45-  48 sec       10.92 Mbits/sec
```

Outside the VPN

PERFORMAN

- PC : iperf -s
- Esp32 : iperf –
- Exceeds PHY sp
  - ~10 Mbit/s ou
  - ~10 Mbit/s in



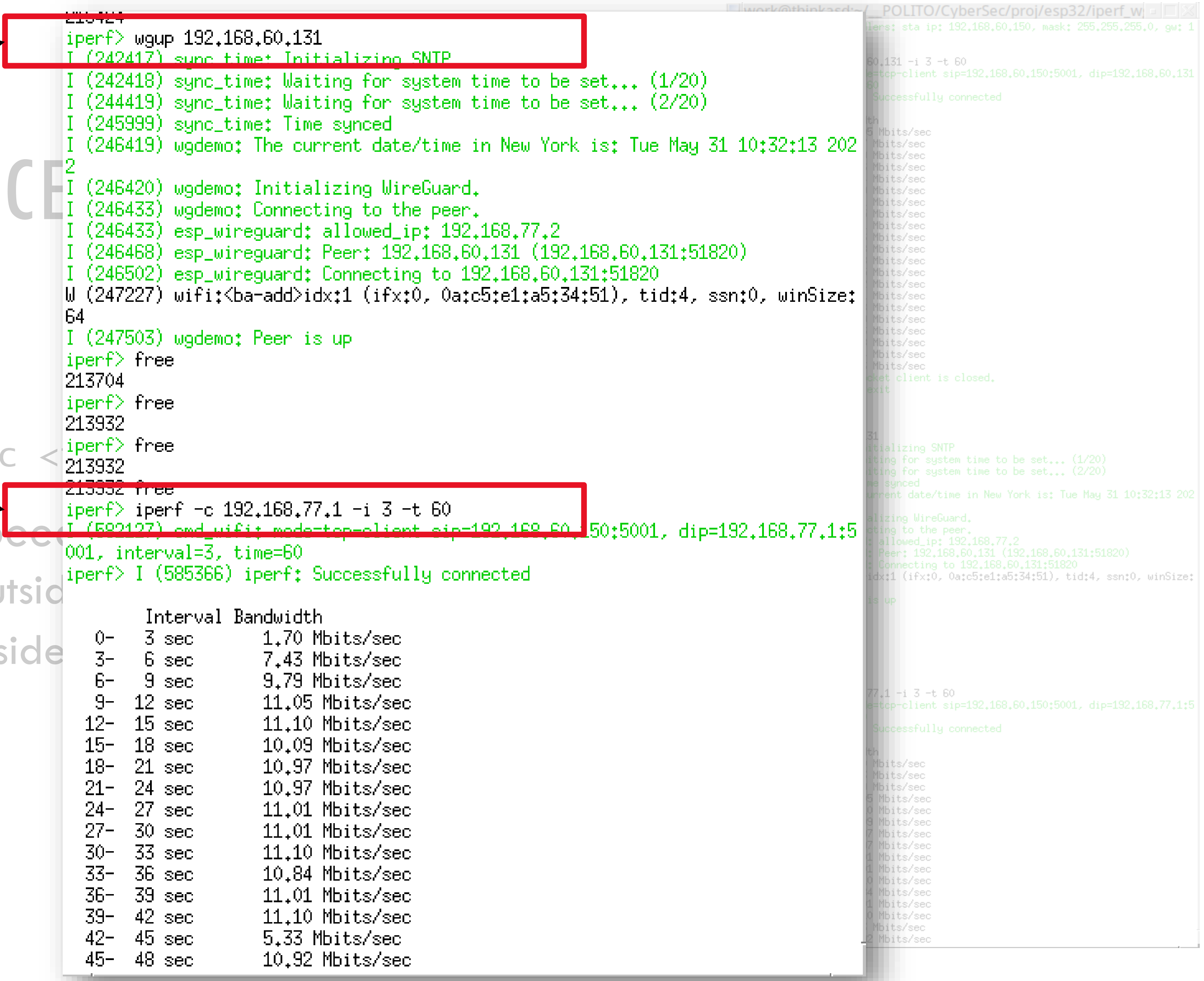
```
iperf>
iperf> iperf -c 192.168.60.131 -i 3 -t 60
I (132052) cmd_wifi: mode=tcp client sip=192.168.60.150:5001, dip=192.168.60.131
:5001, interval=3, time=60
iperf> I (133062) iperf: Successfully connected

        Interval Bandwidth
   0-   3 sec       11.05 Mbits/sec
   3-   6 sec       0.17 Mbits/sec
   6-   9 sec       0.09 Mbits/sec
   9-  12 sec       0.26 Mbits/sec
  12-  15 sec       0.74 Mbits/sec
  15-  18 sec       7.69 Mbits/sec
  18-  21 sec       5.11 Mbits/sec
  21-  24 sec       7.95 Mbits/sec
  24-  27 sec       9.31 Mbits/sec
  27-  30 sec       8.56 Mbits/sec
  30-  33 sec       7.38 Mbits/sec
  33-  36 sec       7.38 Mbits/sec
  36-  39 sec       7.65 Mbits/sec
  39-  42 sec       6.82 Mbits/sec
  42-  45 sec       9.31 Mbits/sec
  45-  48 sec       7.38 Mbits/sec
  48-  51 sec       5.29 Mbits/sec
  51-  54 sec       8.56 Mbits/sec
  54-  57 sec       7.78 Mbits/sec
  57-  60 sec       8.74 Mbits/sec
   0-  60 sec       6.36 Mbits/sec
I (195411) iperf: TCP Socket client is closed.
I (195413) iperf: iperf exit
iperf>
iperf> free
215424
iperf> wgup 192.168.60.131
I (242417) sync_time: Initializing SNTP
I (242418) sync_time: Waiting for system time to be set... (1/20)
```

Starting the VPN

PERFORMANCE

- PC : iperf -s
- Esp32 : iperf -c <
- Exceeds PHY spee
  - ~10 Mbit/s outsid
  - ~10 Mbit/s inside

Inside the VPN



```
iperf> wgup 192.168.60.131
I (242417) sync_time: Initializing SNTP
I (242418) sync_time: Waiting for system time to be set... (1/20)
I (244419) sync_time: Waiting for system time to be set... (2/20)
I (245999) sync_time: Time synced
I (246419) wgdemo: The current date/time in New York is: Tue May 31 10:32:13 202
2
I (246420) wgdemo: Initializing WireGuard.
I (246433) wgdemo: Connecting to the peer.
I (246433) esp_wireguard: allowed_ip: 192.168.77.2
I (246468) esp_wireguard: Peer: 192.168.60.131 (192.168.60.131:51820)
I (246502) esp_wireguard: Connecting to 192.168.60.131:51820
W (247227) wifi:<ba-add>idx:1 (ifx:0, 0a:c5:e1:a5:34:51), tid:4, ssn:0, winSize:
64
I (247503) wgdemo: Peer is up
iperf> free
213704
iperf> free
213932
iperf> free
213932
213932 free
iperf> iperf -c 192.168.77.1 -i 3 -t 60
I (582127) cmd_wifi: mode=tcp-client sip=192.168.60.150:5001, dip=192.168.77.1:5
001, interval=3, time=60
iperf> I (585366) iperf: Successfully connected

        Interval Bandwidth
   0-   3 sec       1.70 Mbits/sec
   3-   6 sec       7.43 Mbits/sec
   6-   9 sec       9.79 Mbits/sec
   9-  12 sec       11.05 Mbits/sec
  12-  15 sec       11.10 Mbits/sec
  15-  18 sec       10.09 Mbits/sec
  18-  21 sec       10.97 Mbits/sec
  21-  24 sec       10.97 Mbits/sec
  24-  27 sec       11.01 Mbits/sec
  27-  30 sec       11.01 Mbits/sec
  30-  33 sec       11.10 Mbits/sec
  33-  36 sec       10.84 Mbits/sec
  36-  39 sec       11.01 Mbits/sec
  39-  42 sec       11.10 Mbits/sec
  42-  45 sec       5.33 Mbits/sec
  45-  48 sec       10.92 Mbits/sec
```

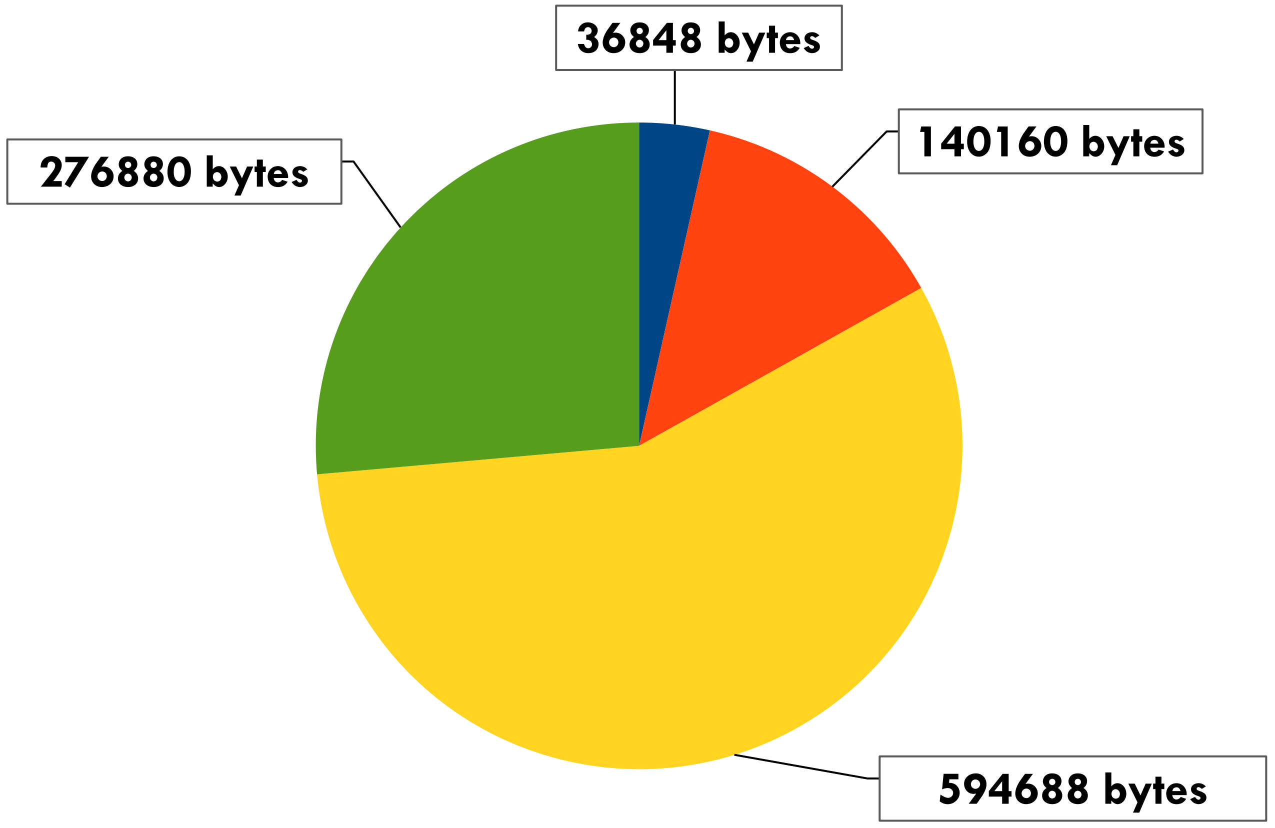
RESOURCES: RAM

(without VPN) - (with VPN) = VPN resources

- Free RAM at IDLE
  - 215424 byte – 213704 byte = 1720 byte
- Free RAM for iperf
  - 19940 byte – 13628 byte = 6312 byte

Note: expect ± 100 byte variations

RESOURCES: FLASH



Flash bytes

Total: 1 048 576 bytes

🟦 Wireguard VPN
🟥 iperf
🟨 FreeRTOS & LwIP
🟩 Free

Note: expect ± 100 byte variations

APPLICATIONS

Pro:
No app modifications are necessary:
VPN module integrates completely in LwIP

Cons:
VPN configuration is still necessary

Use only when needed

- Sensors and Actuators that must **appear inside a company network**
- IoT devices that need to access **services around the world**

THANK YOU

References:

https://github.com/trombik/esp_wireguard

https://www.wireguard.com/papers/wireguard.pdf

Politecnico di Torino