Computer Security Capstone

Project I: IPsec Session Hijacking

Chi-Yu Li (2024 Spring)
Computer Science Department
National Yang Ming Chiao Tung University

Goals

- Understand how to hijack IPsec sessions

- You will learn about
  - Sniffing IPv4/ESP/TCP packets
  - Dumping the key to generate HMAC signature
  - Fabricating IPv4/ESP/TCP packets

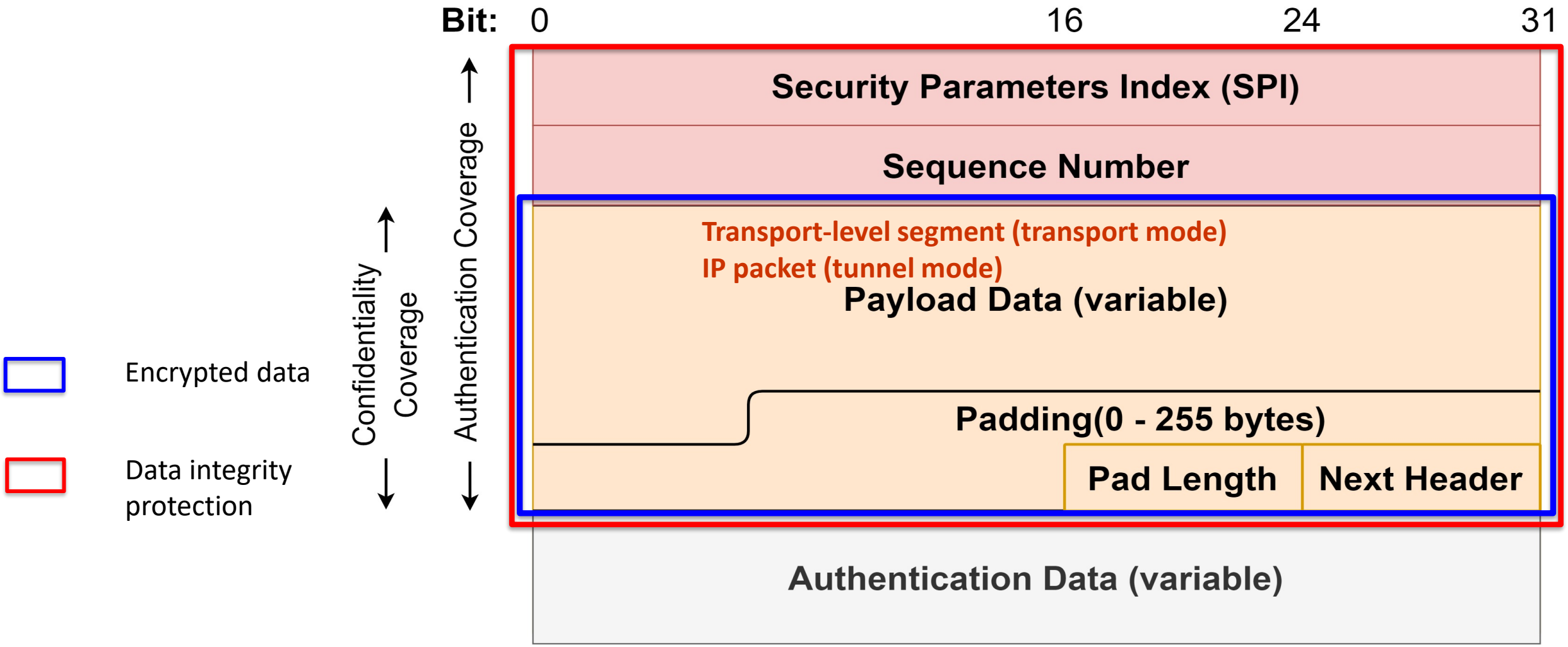
What is IPsec?

- Internet Protocol Security (IPsec) is a secure network protocol suite
  - It provides secure communication by authenticating and encrypting data
  - It ensures the confidentiality and integrity of the data
- Two main protocols
  - Internet Key Exchange (IKE): Used for negotiation and establishment of security associations (SAs)
  - Encapsulating Security Payload (ESP): Provides confidentiality, integrity, authentication

IPsec Primer: Security Associations

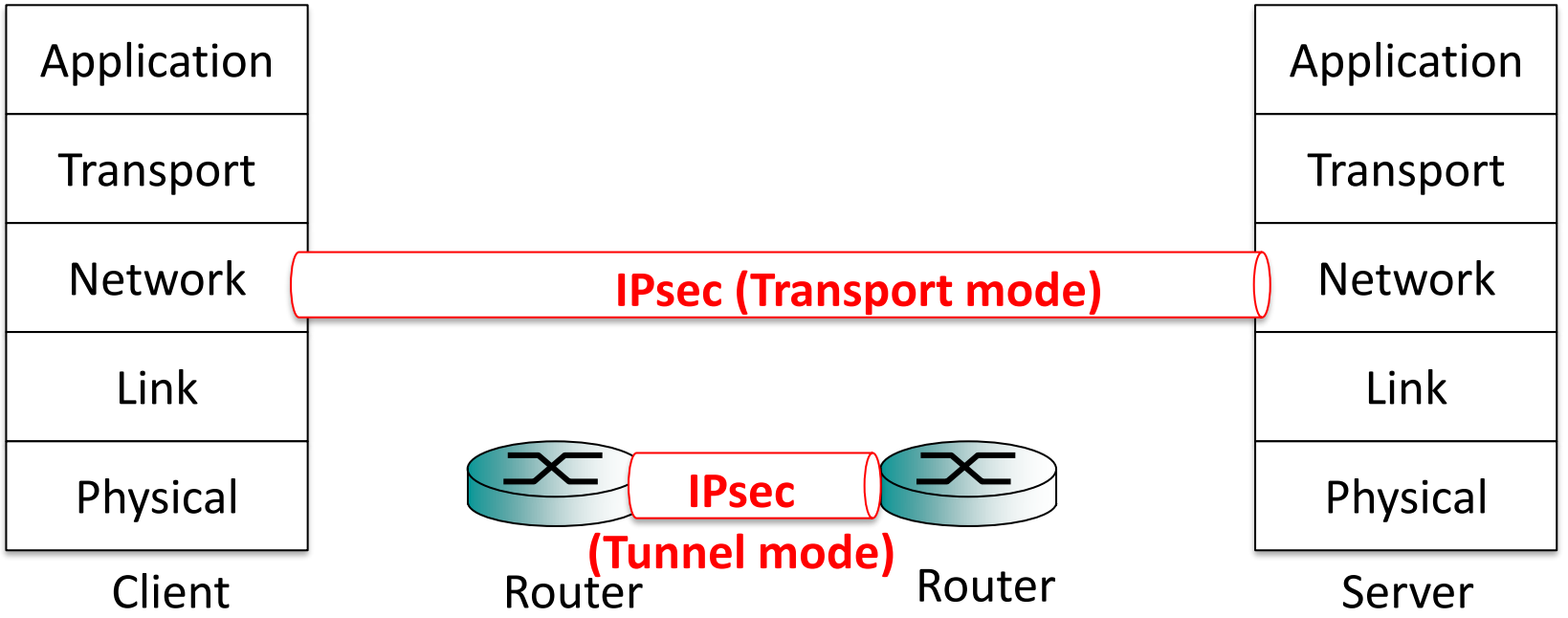
- A key concept of IPsec
  - One SA only governs security for unidirectional traffic
  - Two SAs are necessary for bidirectional traffic
- Uniquely identified by three parameters
  - Security parameter index (SPI)
  - IP destination address
  - Protocol identifier (ESP)

IPsec Primer: Encapsulating Security Payload (ESP)



IPsec Primer: Two IPsec Operation Modes

- Transport and Tunnel modes



IPsec Primer: Transport and Tunnel Modes

Transport Mode

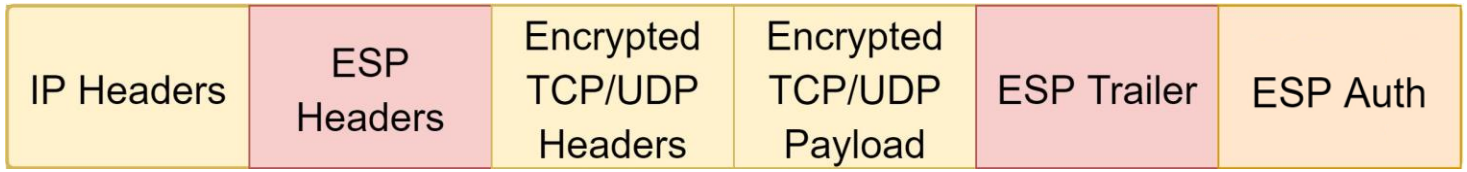
- Protects the payload of the IP packet
- Preserves the original IP header
- Used for end-to-end communication

Tunnel Mode

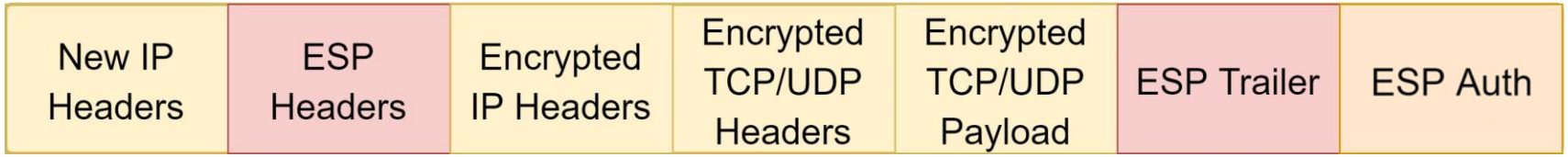
- Protects the entire IP packet
- Adds a new IP header to the packet
- Used for VPNs and gateway-to-gateway communication

IPsec Primer: Data Encapsulation

- ESP Transport mode

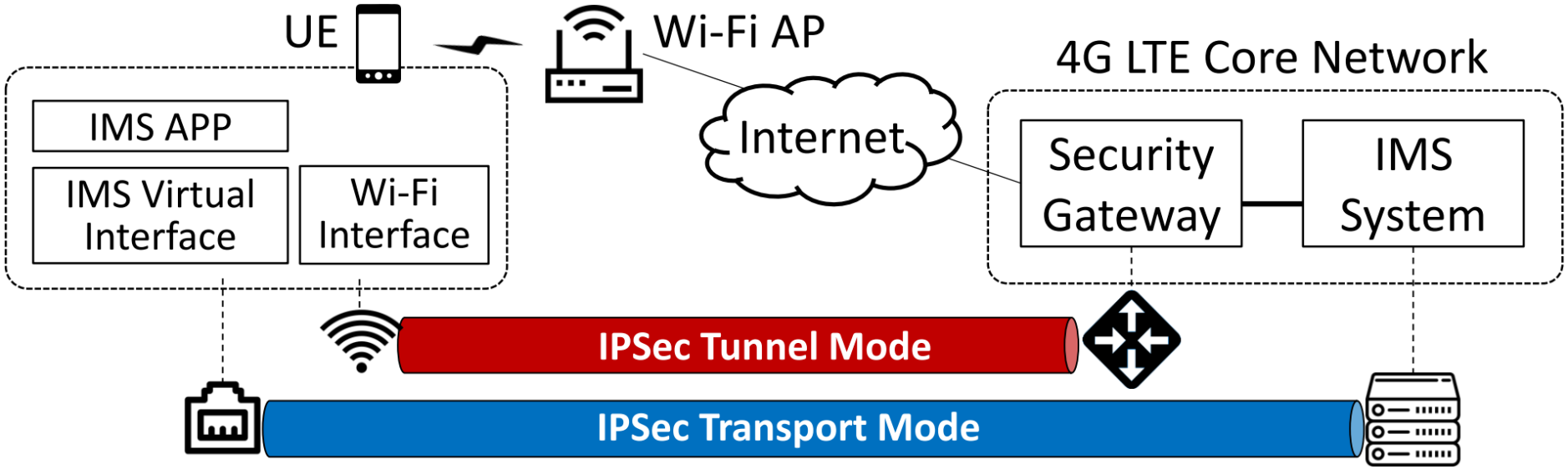


- ESP Tunnel mode



Case Study: VoWi-Fi with IPSec Protection

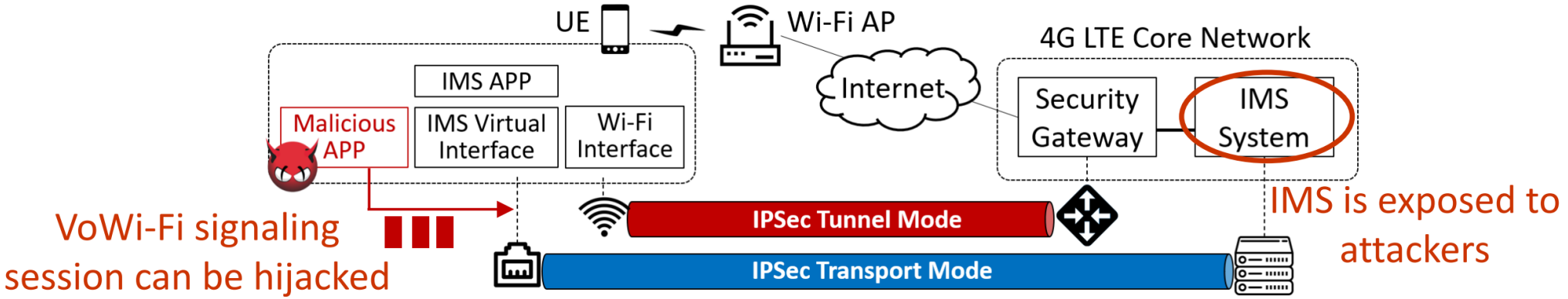
- IPSec protection over VoWi-Fi traffic traversing public domain



Case Study: VoWi-Fi with IPSec Protection (cont.)

- IPSec protection over VoWi-Fi traffic traversing public domain

However, they can be hacked!!

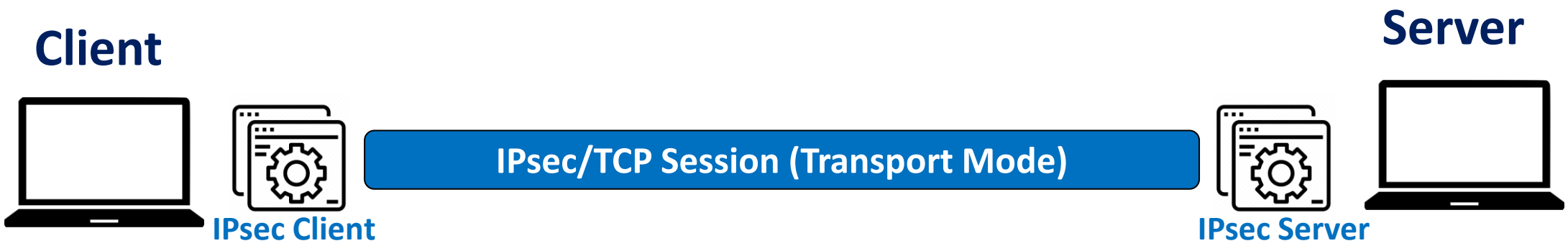


Attacker can manipulate IMS call service operation

IMS vulnerabilities can be exposed!!

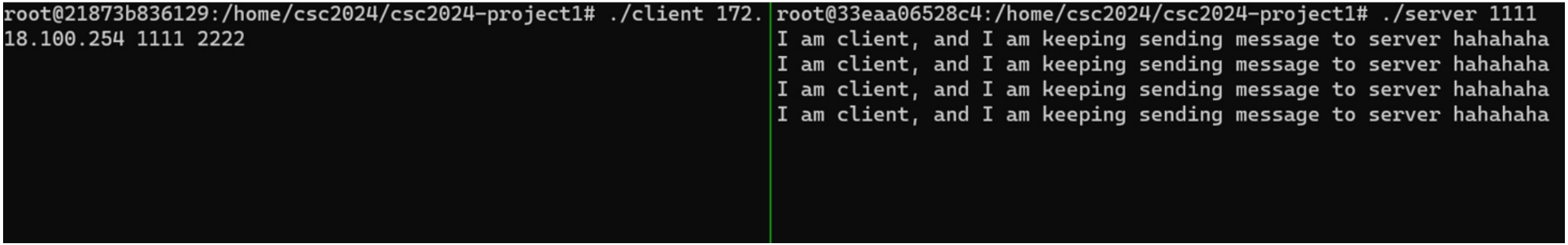
Environment Setup in this project

- Use two docker containers, designated as the client and the server, and establish the IPsec/TCP session between them
  - Please download the supplement from E3
  - Follows the step in the README.md to do the environment setup



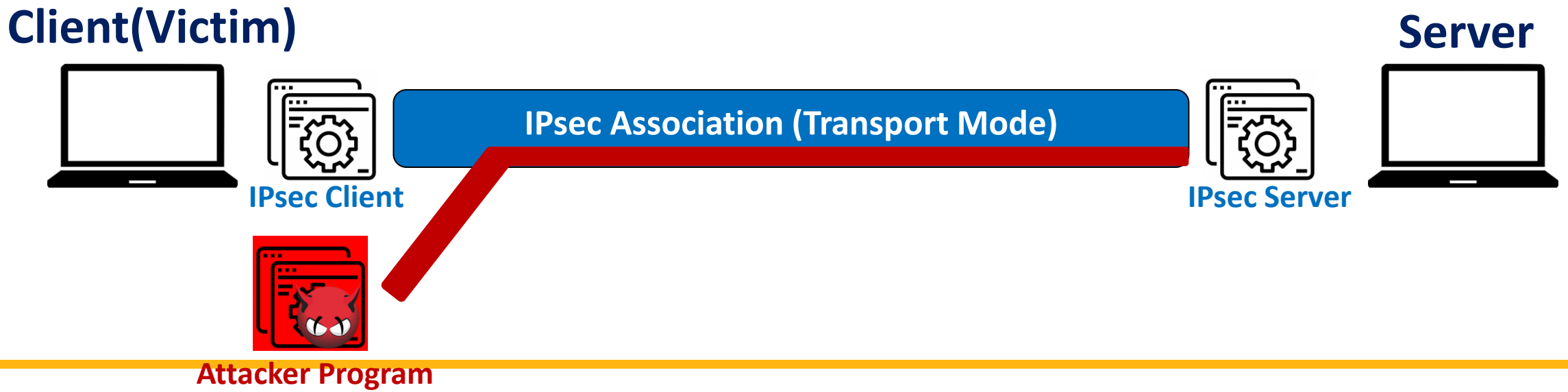
Environment Setup in this project (cont.)

- Use two docker containers, designated as the client and the server, and establish the IPsec/TCP session between them
  - Please download the supplement from E3
  - Follows the step in the README.md to do the environment setup
  - If the setup is successful, the server will keep showing some messages



IPsec/TCP Session Hijacking

- Execute provided programs to establish the IPsec/TCP session
- Develop an attacker program on Client to do hijacking
- Send specific flags to the server using the attacker program
  - If hijacking is successful, the server will reply to flags with correct responses



What should the attacker program do?

- Realtime information monitoring and collecting
  - Get session information from ESP and TCP headers, e.g., ESP SPI and TCP sequence number
  - Retrieve the IPsec SA from security association database (SADB), e.g., ESP secret key
    - Dump the key from SADB (RFC2367 Section 2.3.4 & 2.4 & 3.1.10)
- IPsec/TCP packet crafting
  - Fabricate IPv4/ESP/TCP headers, including all the fields and checksum
  - Generate ESP padding and authentication data
    - The Pad Length and Next Header fields must be right aligned with a 4-byte word (RFC4303 Section 2.4)

Todo Check List for Sample Codes

Method	Description
Session::Session	Fill struct sockaddr_ll addr which will be used to bind the socket
Session::dissect	Set payload
Session::dissectIPv4	Extract IPv4 header and payload, and check if receiving packet from remote
Session::dissectESP	Extract ESP header and payload, and track ESP sequence number
Session::dissectTCP	Extract TCP header, and track TCP header parameters
Session::encapsulateIPv4	Fill IPv4 header, and compute the checksum
Session::encapsulateESP	Fill ESP header, padding, and HMAC parameters
Session::encapsulateTCP	Fill TCP header, and compute the checksum
getConfigFromSADB	Fill struct sadb_msg msg which will be used to create PF_KEY_V2 socket
getConfigFromSADB	Extract SADB information from PF_KEY_V2 socket, and parse them

Three Verification Steps

- Step I: The server can receive fabricated IPsec packets belonging to the existing IPsec session (40%)
- Step II: The attacker program can correctly exchange TCP packets (data and ACK) with the server through the fabricated IPsec packets (30%)
- Step III: The attacker program can interact with the server with multiple handshakes (30%)

Step I: the server can receive fabricated IPsec packets belonging to the existing IPsec session

- An example
  - Client/Attacker program: 172.18.1.1
  - Server: 172.18.100.254

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.18.1.1	172.18.100.254	ESP	142	ESP (SPI=0x0000c6f8)
2	0.000130	172.18.100.254	172.18.1.1	ESP	78	ESP (SPI=0xfb170e3f)
3	0.000198	172.18.1.1	172.18.100.254	ESP	90	ESP (SPI=0x0000c6f8)
4	0.000285	172.18.100.254	172.18.1.1	ESP	78	ESP (SPI=0xfb170e3f)
5	0.000357	172.18.100.254	172.18.1.1	ESP	86	ESP (SPI=0xfb170e3f)
6	0.000628	172.18.1.1	172.18.100.254	ESP	78	ESP (SPI=0x0000c6f8)
7	1.000344	172.18.1.1	172.18.100.254	ESP	142	ESP (SPI=0x0000c6f8)
8	1.041238	172.18.100.254	172.18.1.1	ESP	90	ESP (SPI=0xfb170e3f)

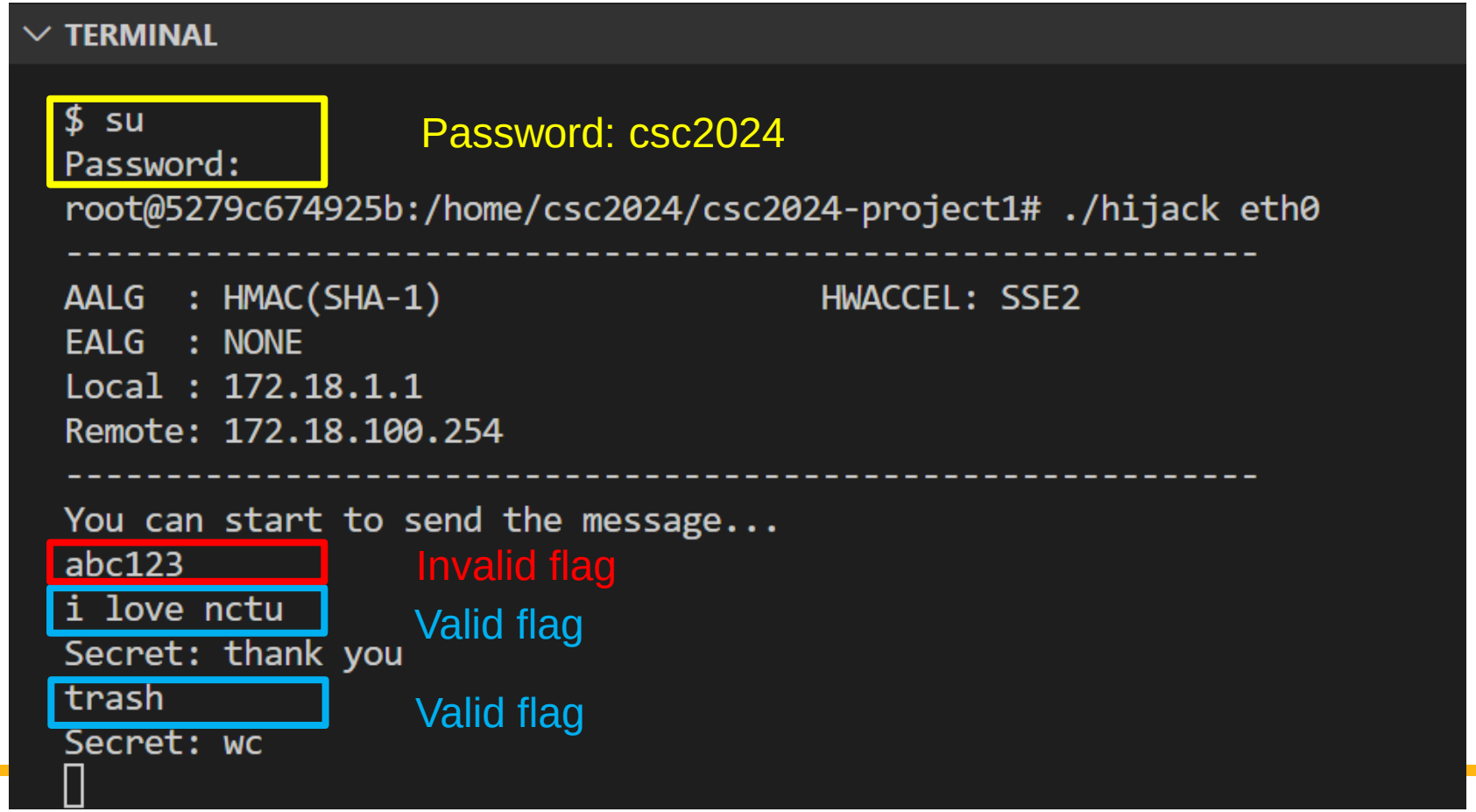
Step II: the attacker program can correctly exchange TCP packets with the server through the fabricated IPsec packets

- An example
  - Client/Attacker program: 172.18.1.1
  - Server: 172.18.100.254
  - Modify the Wireshark preferences to enable dissecting of raw data

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.18.1.1	172.18.100.254	TCP	142	2222 → 1111 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=65
2	0.000130	172.18.100.254	172.18.1.1	TCP	78	1111 → 2222 [ACK] Seq=95 Ack=66 Win=502 Len=0
3	0.000198	172.18.1.1	172.18.100.254	TCP	90	2222 → 1111 [PSH, ACK] Seq=66 Ack=95 Win=502 Len=11
4	0.000285	172.18.100.254	172.18.1.1	TCP	78	1111 → 2222 [ACK] Seq=95 Ack=77 Win=502 Len=0
5	0.000357	172.18.100.254	172.18.1.1	TCP	86	1111 → 2222 [PSH, ACK] Seq=95 Ack=77 Win=502 Len=9
6	0.000628	172.18.1.1	172.18.100.254	TCP	78	2222 → 1111 [ACK] Seq=77 Ack=104 Win=502 Len=0
7	1.000344	172.18.1.1	172.18.100.254	TCP	142	[TCP Retransmission] 2222 → 1111 [PSH, ACK] Seq=66 Ack=1 Win=
8	1.041238	172.18.100.254	172.18.1.1	TCP	90	[TCP ACKed unseen segment] 1111 → 2222 [ACK] Seq=104 Ack=131

Step III: Multiple Handshake Tests with Three Flags

- An example with an invalid flag and two valid flags



Important: How to Prepare Your Attack Program?

- Must provide a **Makefile** which compiles your source codes into one executable file, named **hijack** **(Missing: -20%)**
- Your attacker program shall be run in the container built by provided docker file and docker compose yaml
  - TA may copy your source code file to the TA's container to do some testing
- Must **use given program's framework**, **otherwise no any credit**
- Not recommend you use docker with WSL as the backend because it may not support creating PF_KEY_V2 socket

Project Submission

- Due date: 3/27 11:55pm
- Makeup submission and demo after the final (totally 75 points at most)
- Submission rules
  - Put your source code files into a directory and name it using your student ID(s)
    - If your team has two members, please concatenate your IDs separated by “-”
  - Zip the directory and upload the zip file to E3
  - A sample of the zip file: 01212112-02121221.zip, you can use “zipinfo -1” to examine the format (Wrong file name or format will result in 10 points deduction)
    - 01212112-02121221/
    - 01212112-02121221/Makefile
    - …
- Teamwork is allowed, up to two members for each team

Project Demo

- Demo date: 3/29
- TA will prepare two containers to run as the client and the server, respectively
  - Your zip file will be put into the client
- You will
  - be asked to launch an IPsec/TCP hijacking attack
  - be only allowed to “make” to compile all your files, and run your attack binary programs or scripts
  - be not allowed to modify your codes or scripts in the demo
  - be not allowed to install any programs or libraries in the container
  - be asked some questions
  - be responsible to show the outcome to TA and explain why you have successfully achieved the goals

Hint 1: Dump the key from SADB

- The message format in SADB

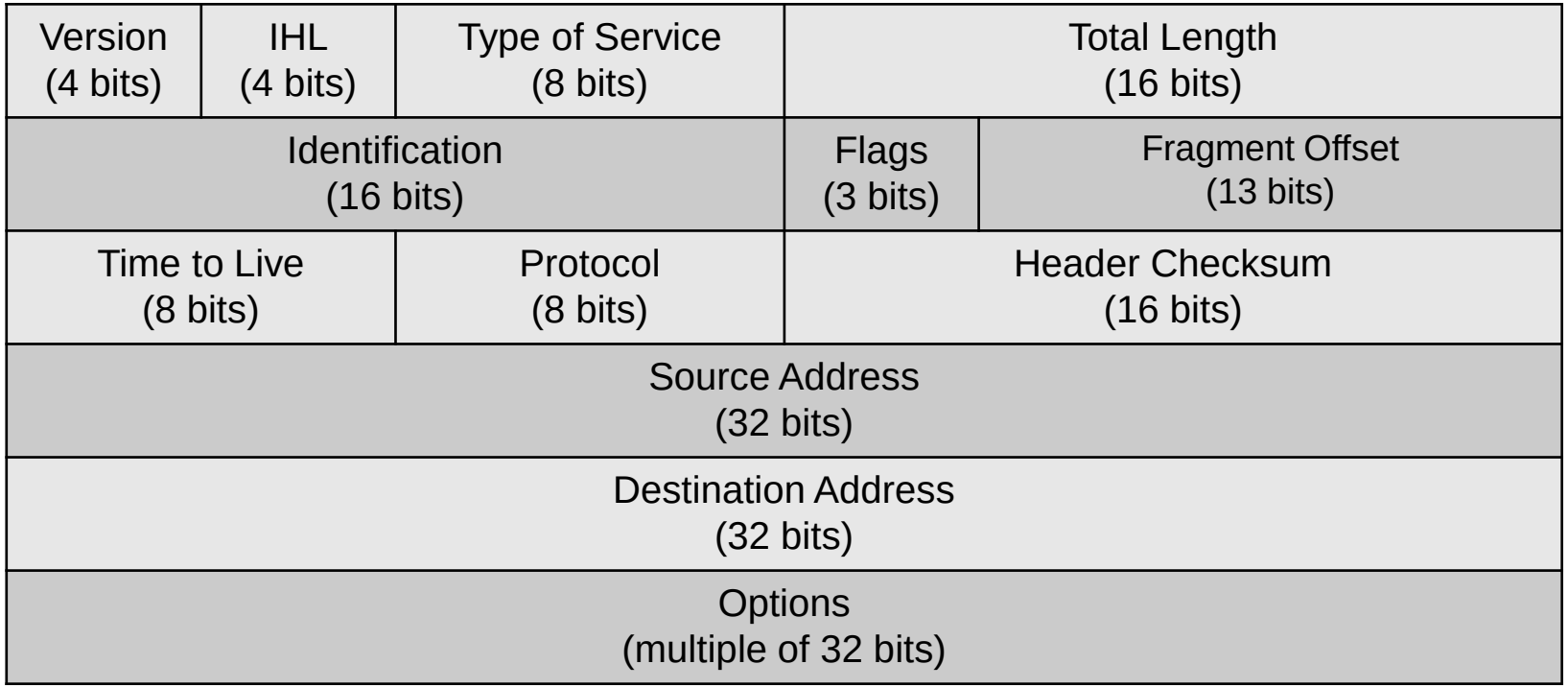
struct sadb_msg	sadb_ext	sadb_ext	…

- Each extension begins with a 16-bit ext_len and a 16-bit ext_type field

- Getting the key from the extension with sadb_ext_type "SADB_EXT_KEY_AUTH"

Hint 2: Fabricate packets

- IPv4 header format



Hint 2: Fabricate packets (cont.)

- ESP format

<table>
<tr><td colspan="3">Security Parameter Index (SPI)
(32 bits)</td></tr>
<tr><td colspan="3">Sequence Number
(32 bits)</td></tr>
<tr><td colspan="3">ESP Payload Data</td></tr>
<tr><td>ESP Payload Data</td><td colspan="2">Padding</td></tr>
<tr><td>Padding</td><td>Pad Length
(8 bits)</td><td>Next Header
(8 bits)</td></tr>
<tr><td colspan="3">ESP Authentication Data</td></tr>
</table>

Hint 2: Fabricate packets (cont.)

- TCP header format

<table>
<tr><td colspan="8">Source Port
(16 bits)</td><td colspan="8">Destination Port
(16 bits)</td></tr>
<tr><td colspan="16">Sequence Number
(32 bits)</td></tr>
<tr><td colspan="16">Acknowledge Number
(32 bits)</td></tr>
<tr><td colspan="4">Header Length
(4 bits)</td><td colspan="4">Reserved Bits
(6 bits)</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="8">Window Size
(16 bits)</td></tr>
<tr><td colspan="8">Checksum
(16 bits)</td><td colspan="8">Urgent Pointer
(16 bits)</td></tr>
<tr><td colspan="16">Options</td></tr>
</table>

Questions?