Encryption ≠ Security

Just because data is encrypted doesn’t mean it is secure.

Lab2

Encrypting Everything

Remember

There is no 100% security

Security, like all engineering, involves tradeoffs

Know what you are trying to secure

The adversary…

State Sponsored

Question from last time…

How can a standardized protocol be secure if the adversary can buy the standard?

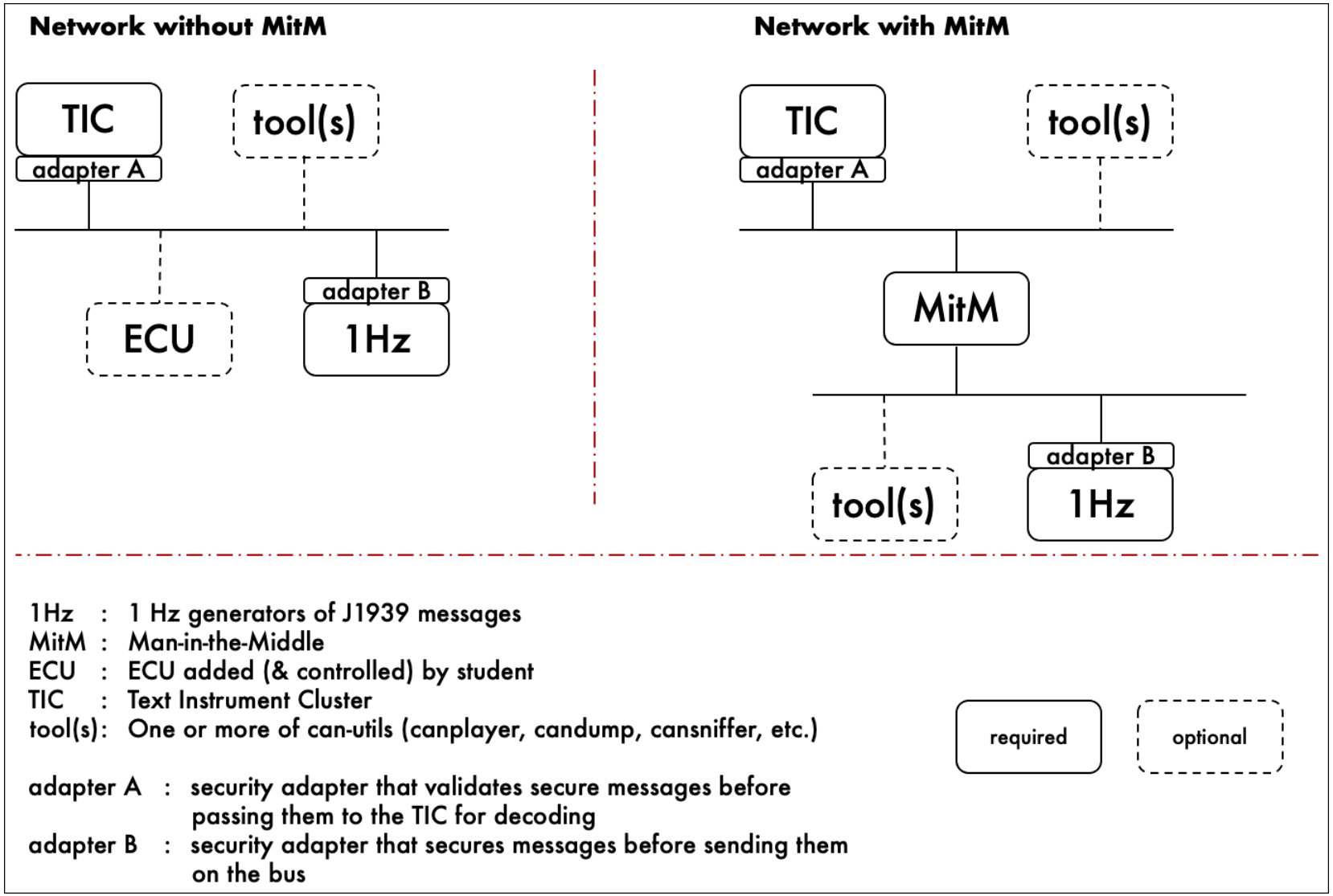
Kerckhoffs's principle

> The principle holds that a cryptosystem should be secure, even if everything about the system, except the key, is public knowledge. This concept is widely embraced by cryptographers, in contrast to security through obscurity, which is not.

Found in: *La Cryptographie Militaire, 1883*

Network Configuration

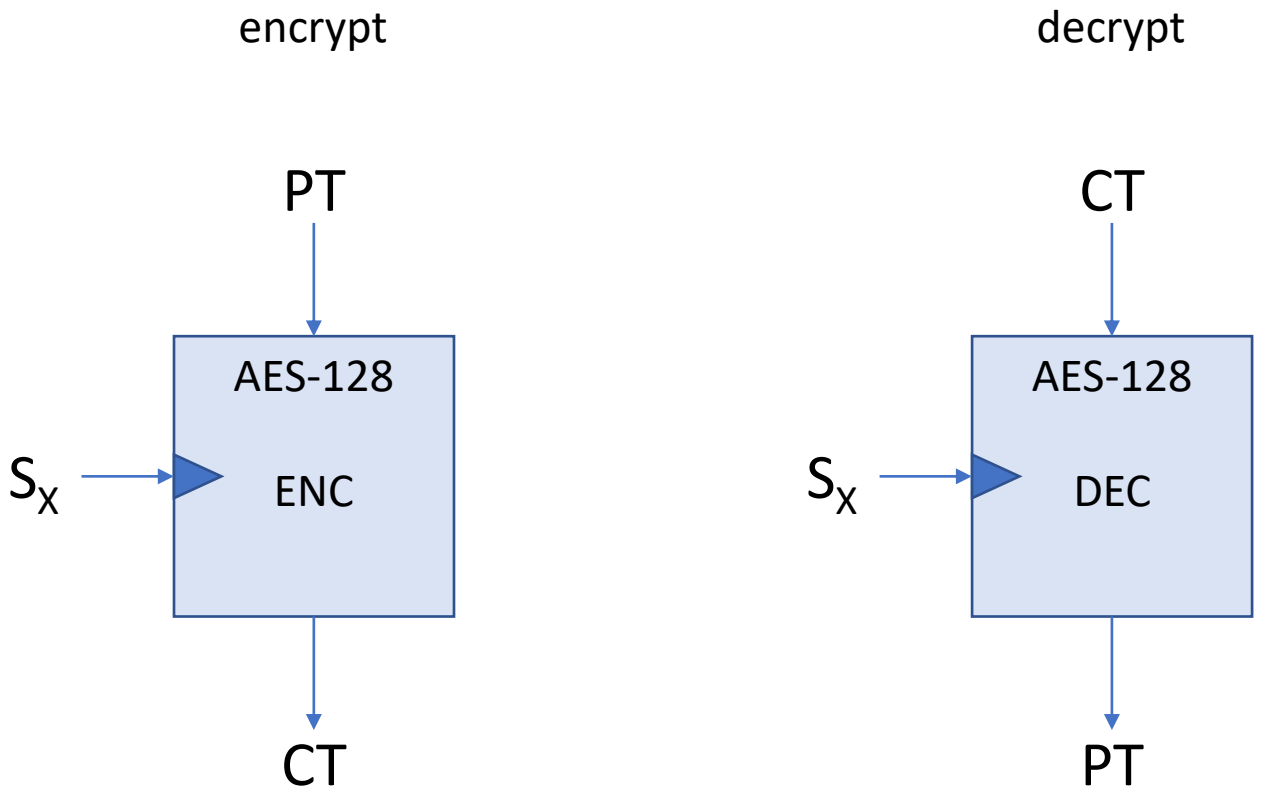
Use both configurations in this lab



Historical Reference

- Personal experience – brainstorming sessions early in the development of product security
  - Make it “easy”, “just encrypt everything”
  - “128-bit encryption is unbreakable”

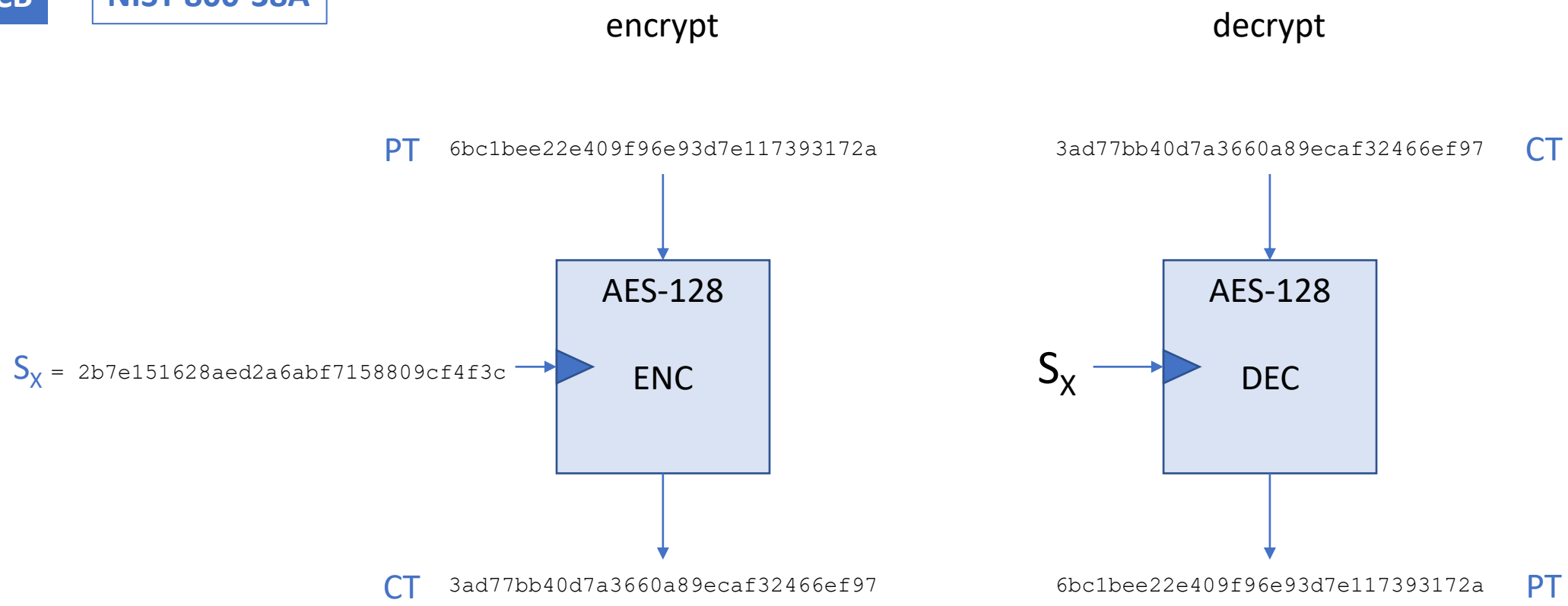
AES-128 ECB



S_X: symmetric key for entity “x”
PT: plaintext
CT: ciphertext

ECB: electronic codebook mode

AES-128 ECB **NIST 800-38A**



https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

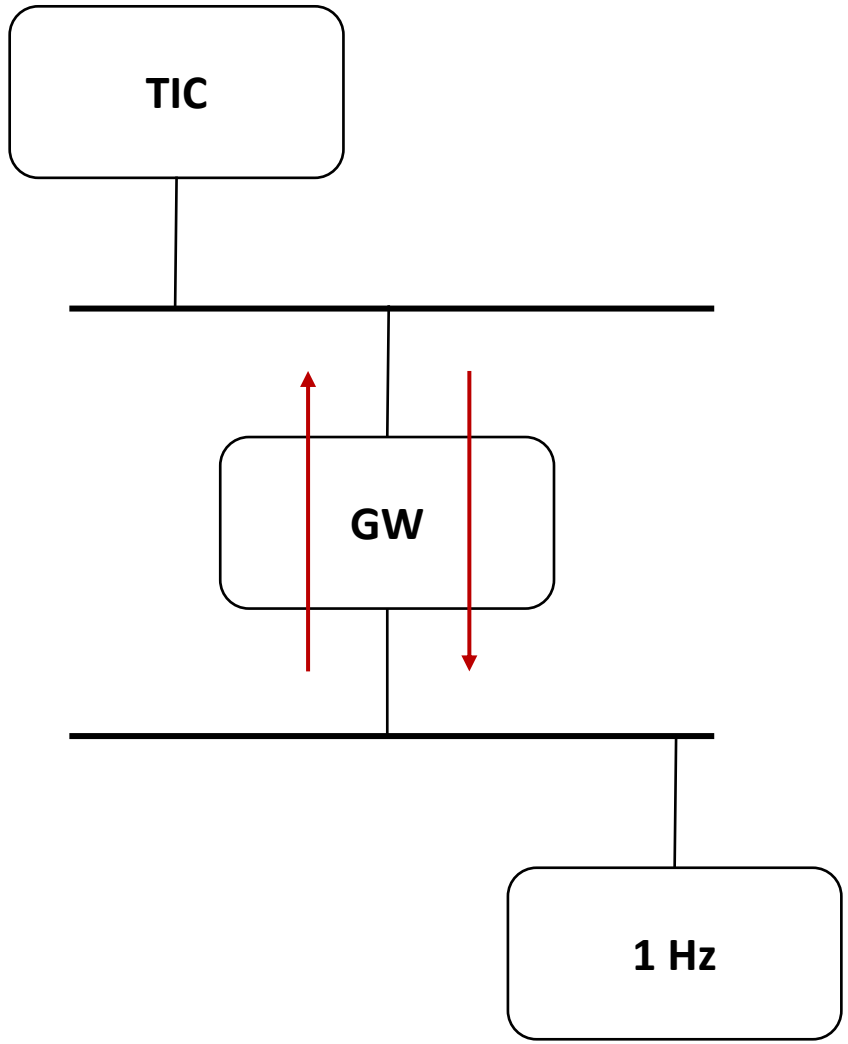
ECB example

```
1  # https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/
2  # https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/#interfaces
3
4  import os
5  from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
6
7  key = os.urandom(16)
8
9  # https://docs.python.org/3/library/stdtypes.html#bytes-objects
10
11 # https://docs.python.org/3/library/stdtypes.html#bytes.fromhex
12 # NOTE can use spaces and size the bytes as needed
13 key2 = bytes.fromhex("00000000 11111111 22222222 33333333")
14
15 # https://docs.python.org/3/library/stdtypes.html#bytes.hex
16 # NOTE can choose a seperator and number of bytes per seperator
17 print(key.hex(" ",4))
18 print(key2.hex(" ",4))
19
20 cipher = Cipher(algorithms.AES(key), modes.ECB())
21
22 encryptor = cipher.encryptor()
23 pt = b"0123456789abcdef"
24 ct = encryptor.update(pt) + encryptor.finalize()
25
26 decryptor = cipher.decryptor()
27 pt2 = decryptor.update(ct) + decryptor.finalize()
28
29 print(pt)
30 print(pt2)
```

```
> python3 ecb.py | cat -n
     1  c6a9c747 691204c7 22bd76cd 7821a3a2
     2  00000000 11111111 22222222 33333333
     3  b'0123456789abcdef'
     4  b'0123456789abcdef'
```

Gateway (“gw”) example

Traffic can flow both ways through gateway



Gateway (“gw”) example

```
148     bus0 = can.Bus(interface="socketcan", channel=port0)
149     bus1 = can.Bus(interface="socketcan", channel=port1)
150
151     if dolog:
152       f = open(filename, "w")
153
154     # port1 listeners ..........................................
155     reader1 = can.AsyncBufferedReader()
156
157     # Listeners are explained in [rtd]/listeners.html
158     listeners1: List[MessageRecipient] = [
159         reader1,        # AsyncBufferedReader() listener
160         fwd_1to0,       # Callback function
161     ]
162
163     # port0 listeners ..........................................
164     reader0 = can.AsyncBufferedReader()
165
166     # Listeners are explained in [rtd]/listeners.html
167     listeners0: List[MessageRecipient] = [
168         reader0,        # AsyncBufferedReader() listener
169         fwd_0to1,       # Callback function
170     ]
171
172     # Create Notifier with an explicit loop to use for scheduling of callbacks
173     #   notifier is used as a message distributor for a bus. Notifier
174     #   creates a thread to read messages from the bus and distributes
175     #   them to listeners. [rtd]/api.html#notifier
176     loop1 = asyncio.get_running_loop()
177     notifier1 = can.Notifier(bus1, listeners1, loop=loop1)
178     loop0 = asyncio.get_running_loop()
179     notifier0 = can.Notifier(bus0, listeners0, loop=loop0)
```

2 bus interfaces

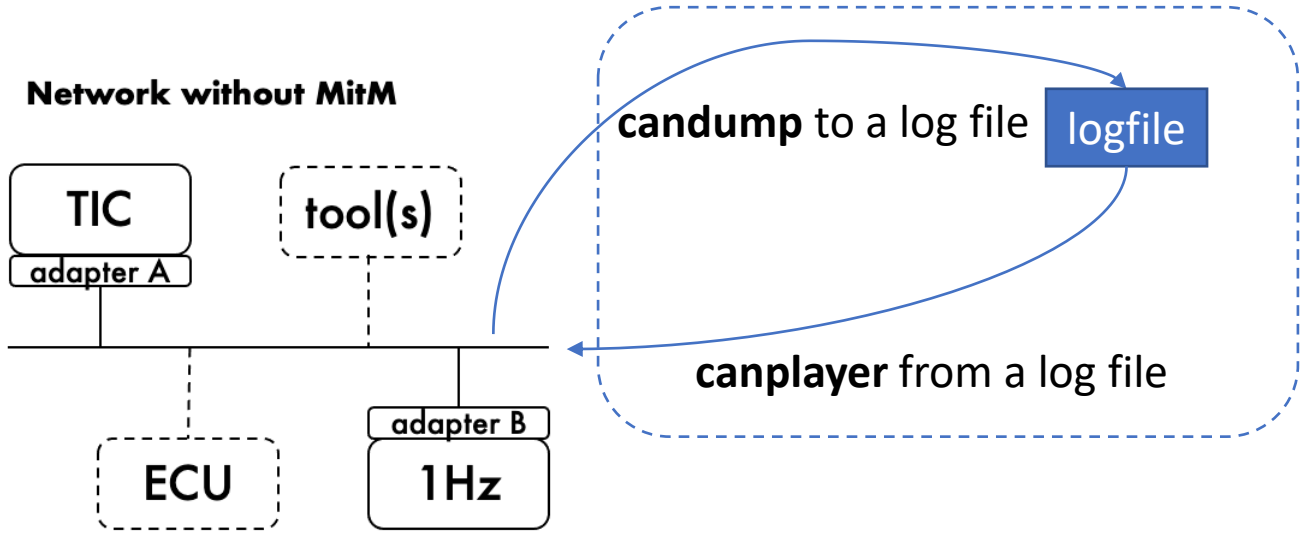
notifiers for each

Gateway (“gw”) example

```
100  def fwd_0to1(msg: can.Message) -> None:
101      global bus1
102      msg = doublespeed(msg)
103      bus1.send(msg)
104      write_message(msg)
105
106  def fwd_1to0(msg: can.Message) -> None:
107      global bus0
108      msg = doublespeed(msg)
109      bus0.send(msg)
110      write_message(msg)
```

In this example the gateway is modifying the message before sending onto the other bus.

Tools: candump & canplayer



Tools: candump & canplayer

```
$ candump -l vcan0

Disabled standard output while logging.

Enabling Logfile 'candump-2023-01-30_104212.log’



$ canplayer -li -I ~/tmp/candump-2023-01-30_104212.log
```

Use manual page (`$ man canplayer`) to find

1. How to work with recording on one interface and playing on another interface
2. [optional] how to tweak replay speed attack.

Lab

- Use S_U = 00000000 11111111 22222222 33333333

For “less noisy attack”

- We don’t have “bootrom” mode in our 1Hz generators
- However, we do make it easy to add a MitM

Two “launchers”

- For exercise 3 you will have to launch the gateway yourself