ShadowAuth: Backward-Compatible Automatic CAN Authentication for Legacy ECUs

Sungwoo Kim, Gisu Yeo, Taegyu Kim, Junghwan "John" Rhee, Yuseok Jeon,
Antonio Bianchi, Dongyan Xu, and Dave (Jing) Tian

PURDUE UNIVERSITY

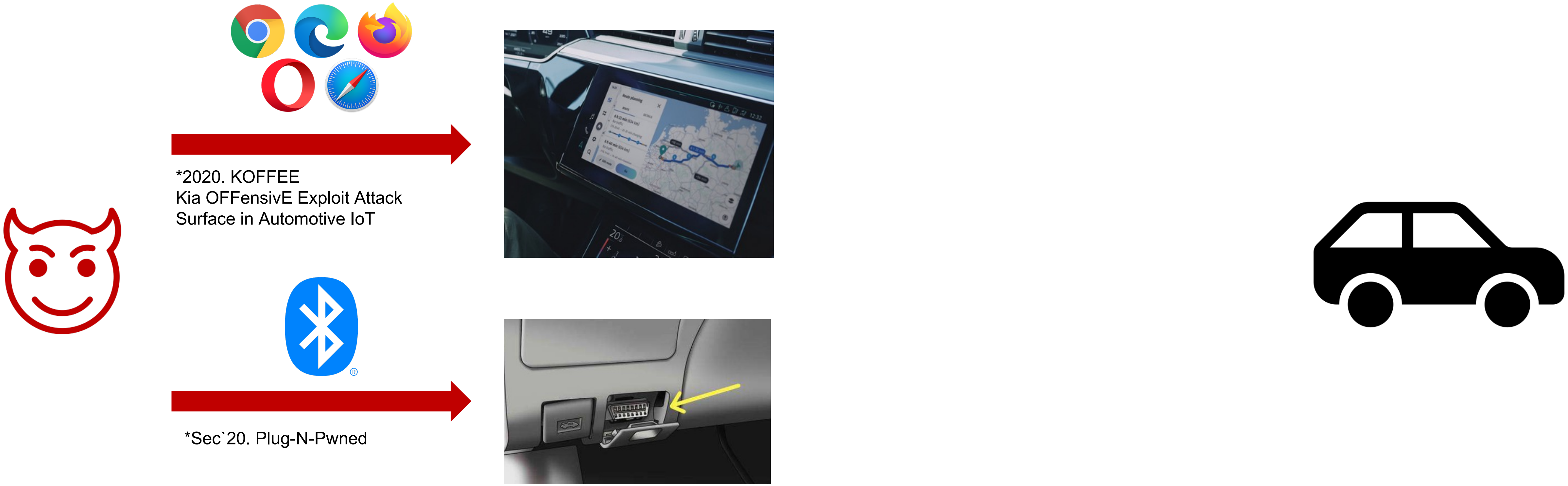
부산대학교
PUSAN NATIONAL UNIVERSITY

PennState

UNIVERSITY OF Central Oklahoma

UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

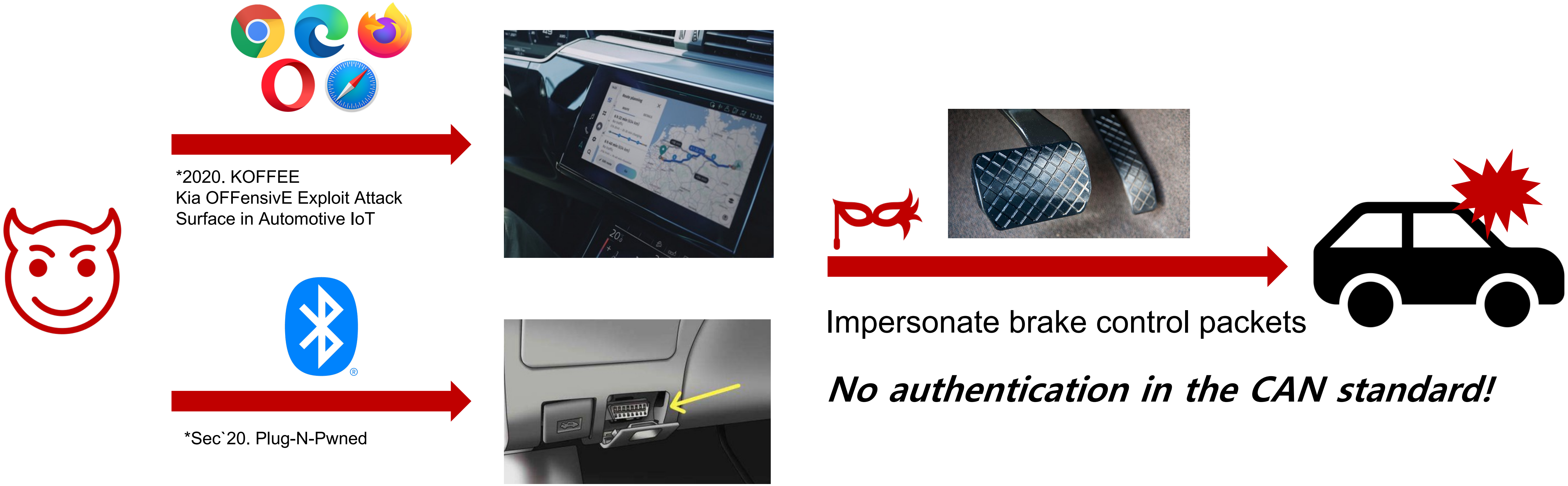
Remote Attacks on In-vehicle Network



*2020. KOFFEE
Kia OFFensivE Exploit Attack
Surface in Automotive IoT

*Sec`20. Plug-N-Pwned

Remote Attacks on In-vehicle Network



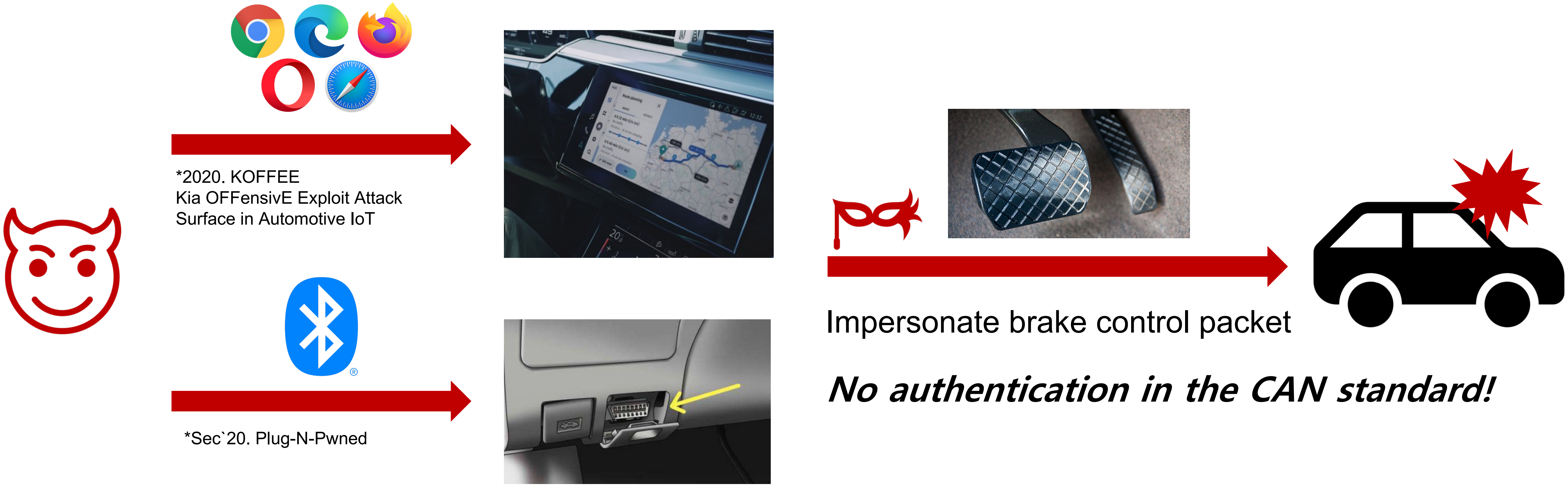
*2020. KOFFEE
Kia OFFensivE Exploit Attack
Surface in Automotive IoT

*Sec`20. Plug-N-Pwned

Impersonate brake control packets

No authentication in the CAN standard!

Remote Attacks on In-vehicle Network



*2020. KOFFEE
Kia OFFensivE Exploit Attack
Surface in Automotive IoT

*Sec`20. Plug-N-Pwned

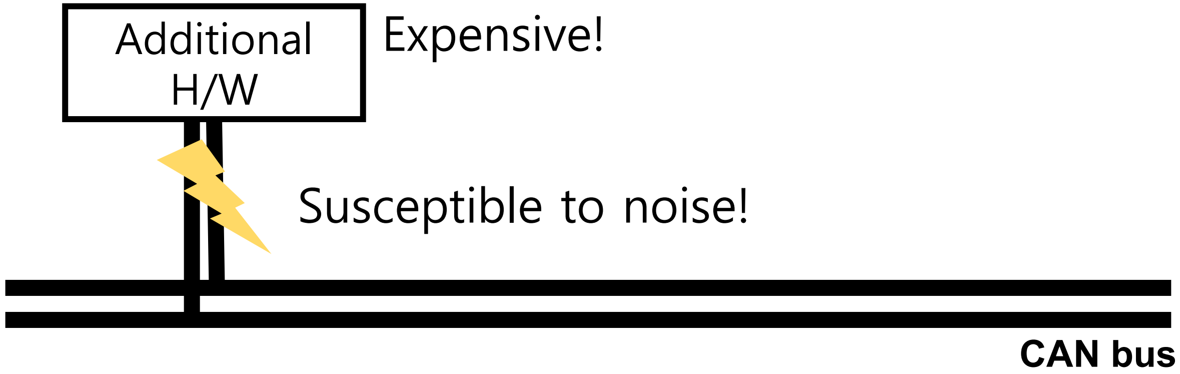
Impersonate brake control packet

No authentication in the CAN standard!

- Previous works proposed authentication feature, but…
- Why has no one **not been deployed** in real-world?

Previous works

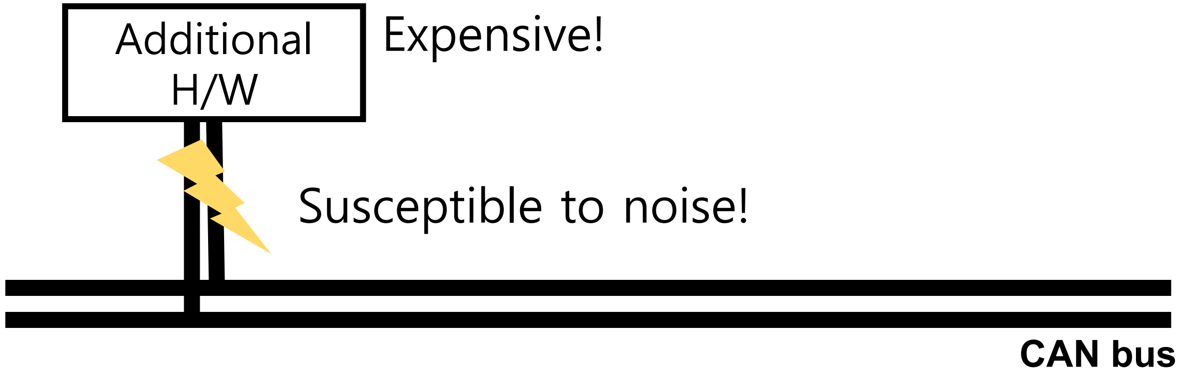
- Side channel-based IDS
  - Additional H/W
  - Not reliable
  - *VIDEN, EASI, CIDS, …



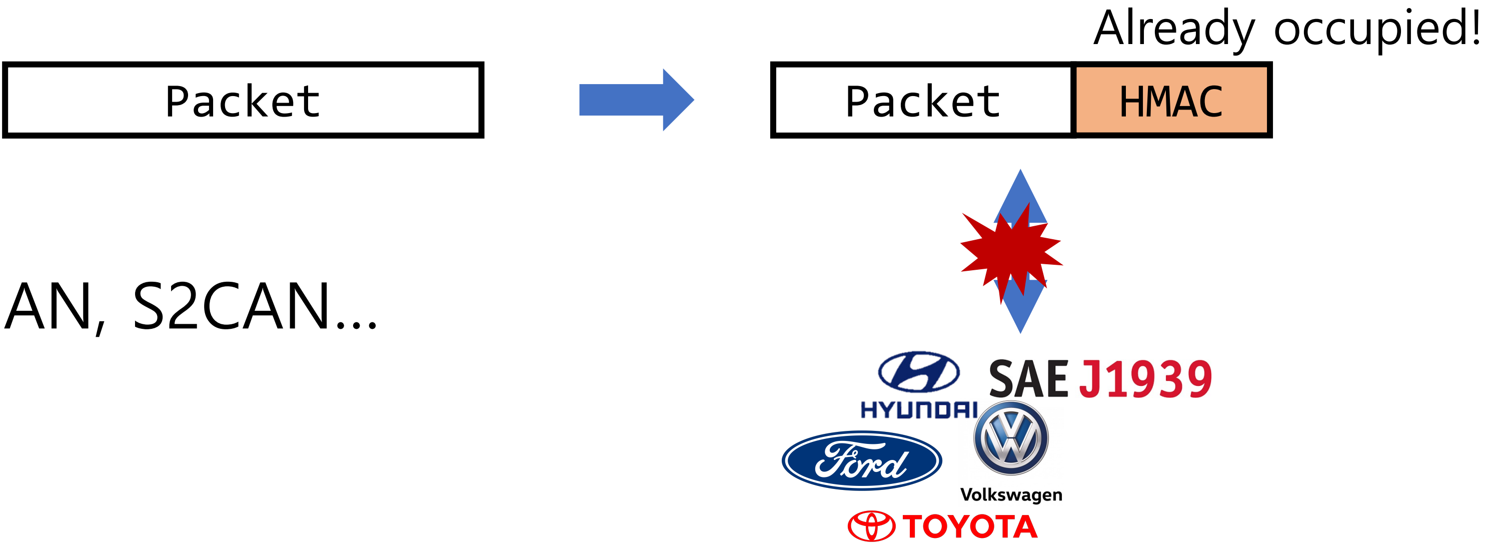
* CCS`17, USENIX`16, NDSS`20

Previous works

- Side channel-based IDS
  - Additional H/W
  - Not reliable
  - VIDEN, EASI, CIDS, …



- HMAC-based IDS
  - Incompatible new packet
  - Additional delay
  - *CANAuth, TOUCAN, LiBRA-CAN, S2CAN…



* ECRYPT`11, AUTOSEC`19, CNS`12, ACSAC `21

Design Goals

- Backward-compatibility
- Accuracy
- No extra delay
- No extra H/W

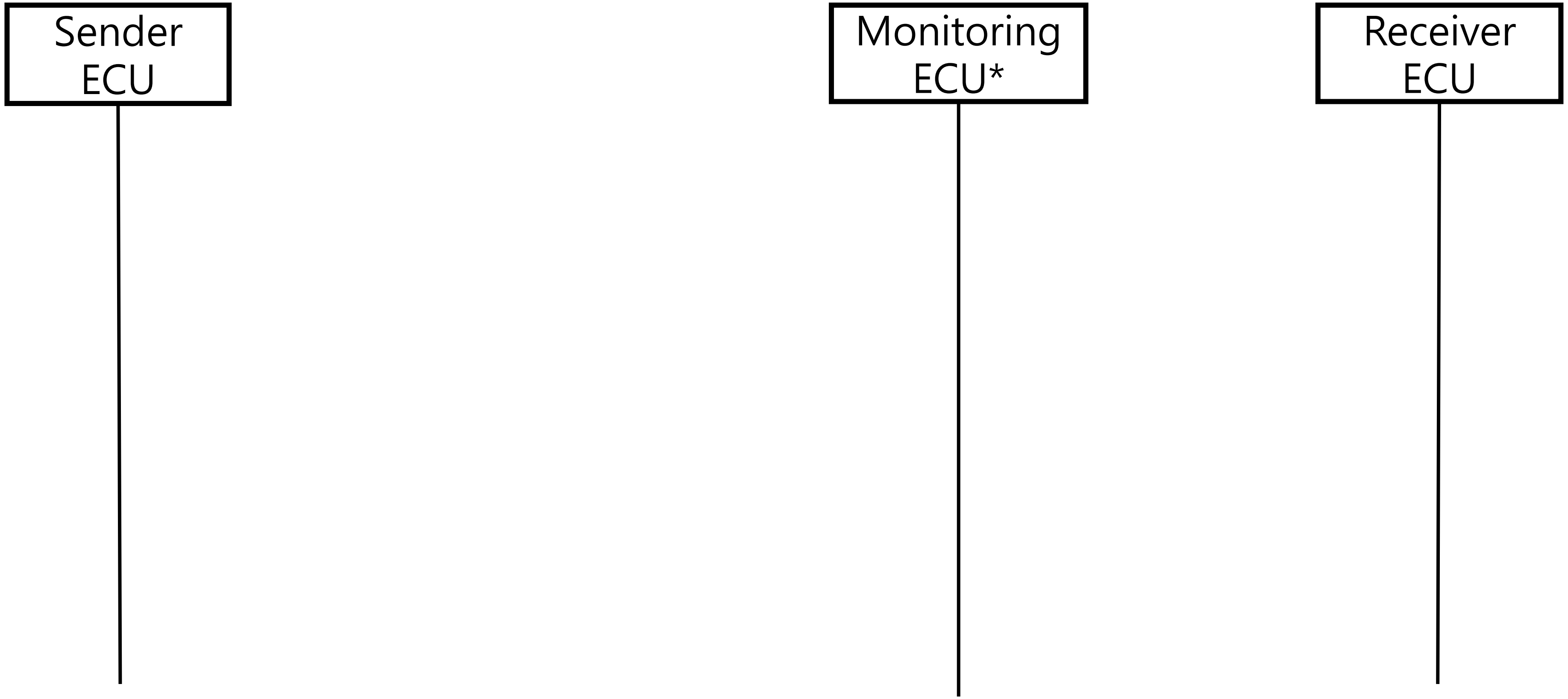
Our Solution: ShadowAuth

- Flexible authentication packets
- HMAC
- Asynchronous authentication
- Binary patching

"Accept-first-authenticate-later"

ShadowAuth Design

ShadowAuth Design



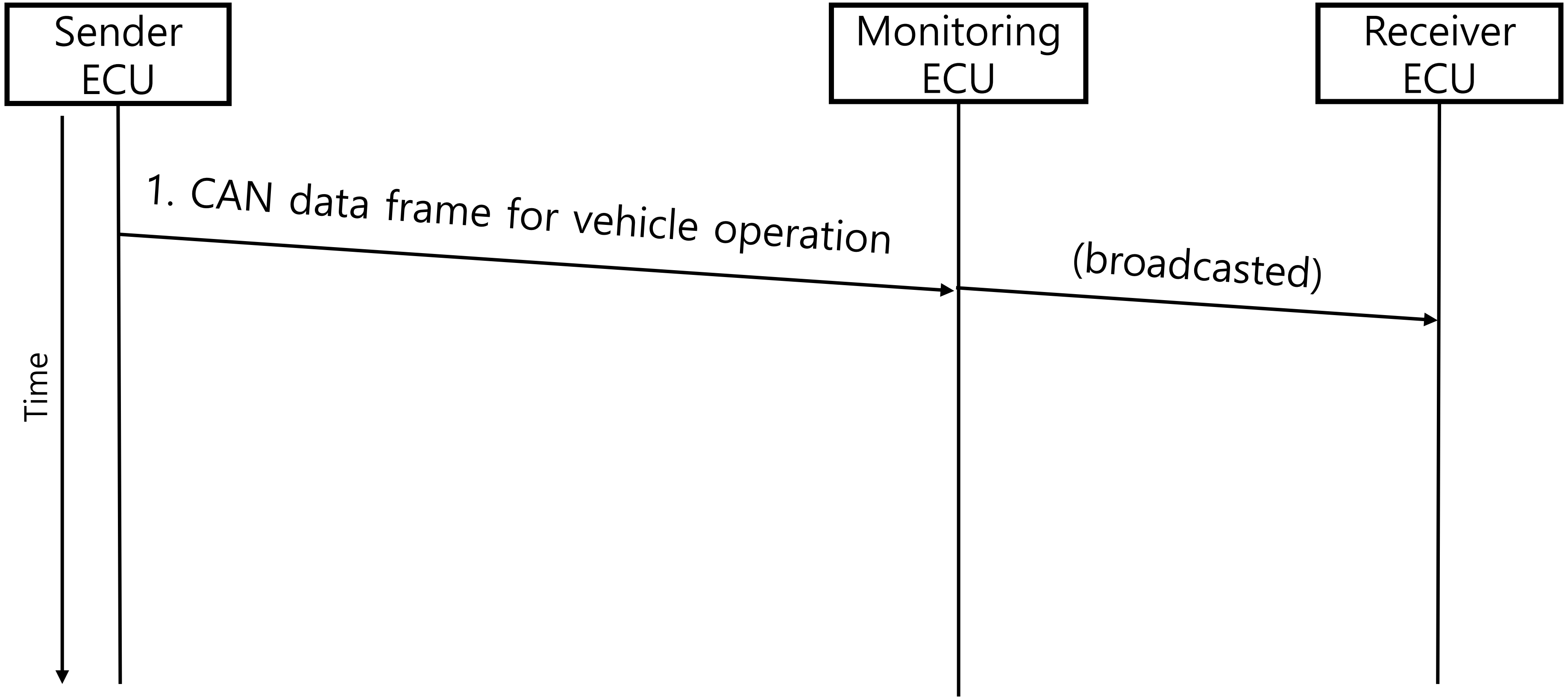
Sender ECU

Monitoring ECU*

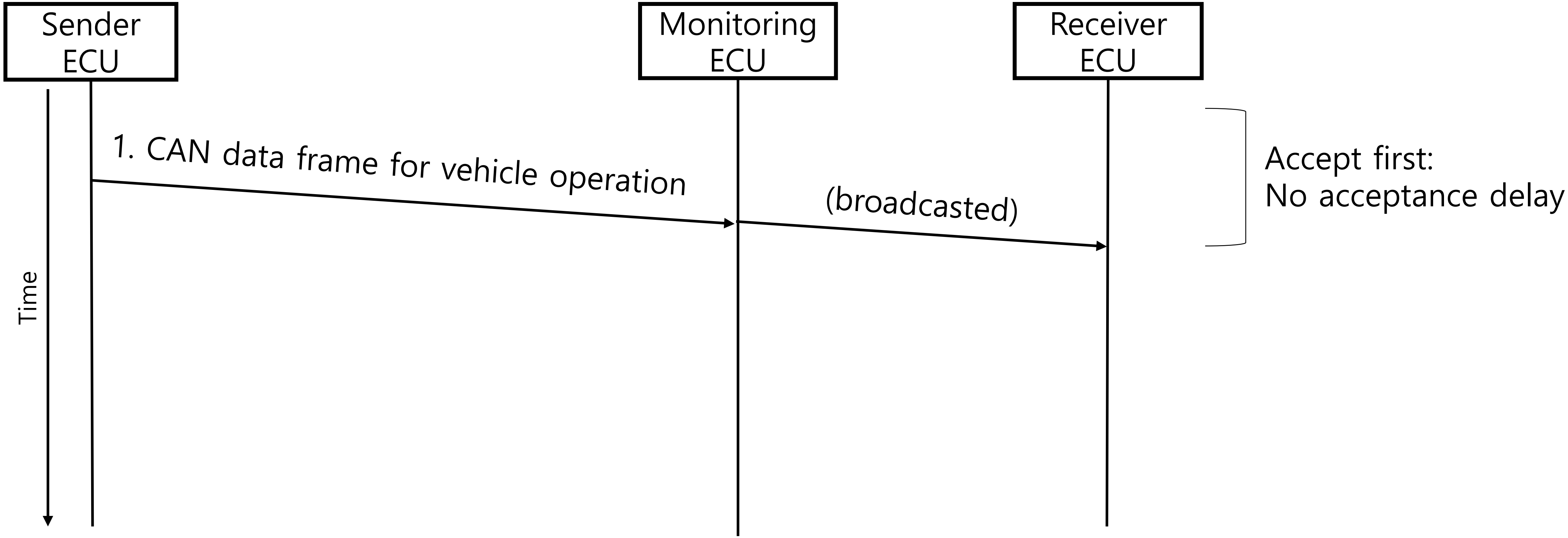
Receiver ECU

*E.g., Gateway ECUs

ShadowAuth Design



ShadowAuth Design



Sender ECU

Monitoring ECU

Receiver ECU

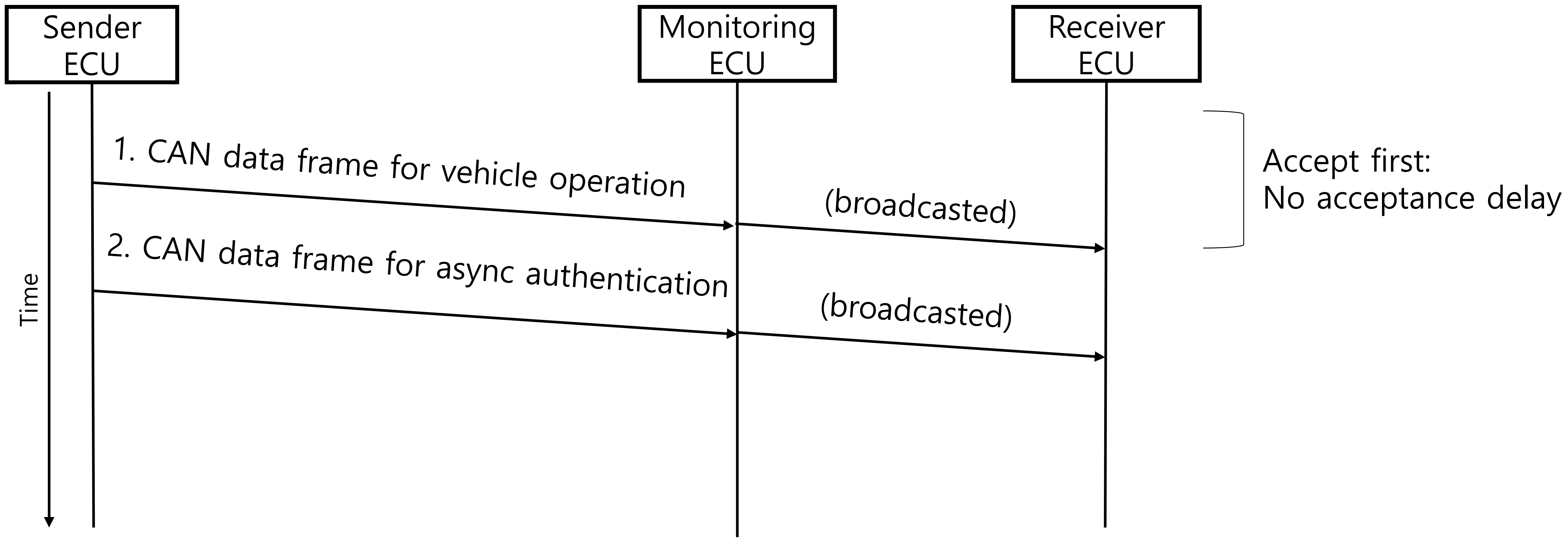
Time

1. CAN data frame for vehicle operation

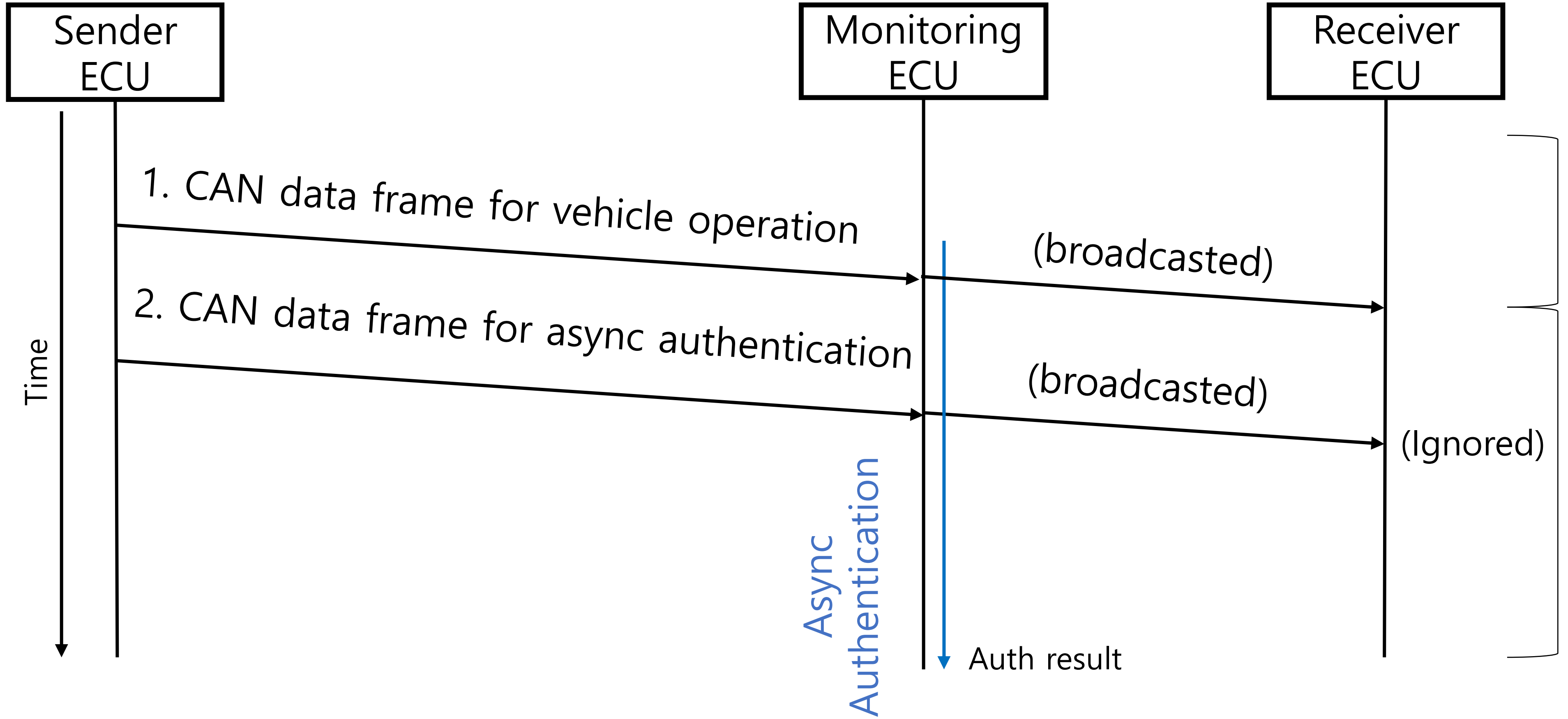
(broadcasted)

Accept first:
No acceptance delay

ShadowAuth Design



ShadowAuth Design

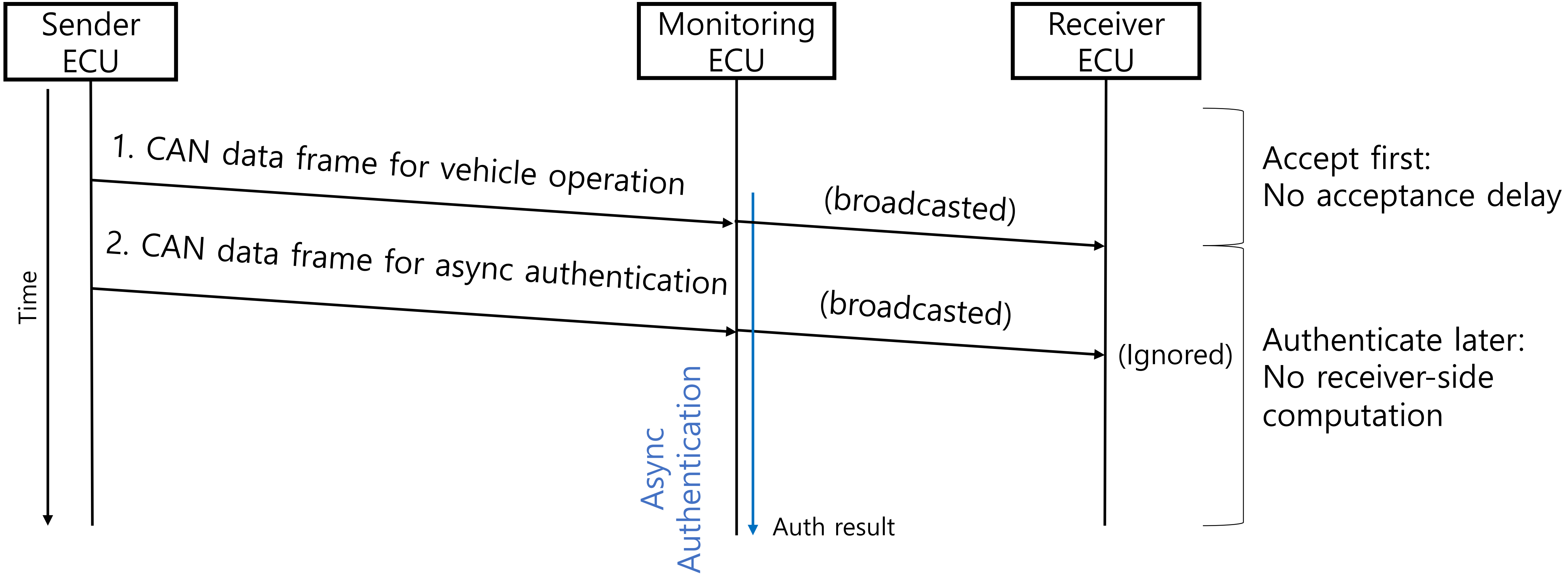


Accept first:
No acceptance delay

Authenticate later:
No receiver-side computation

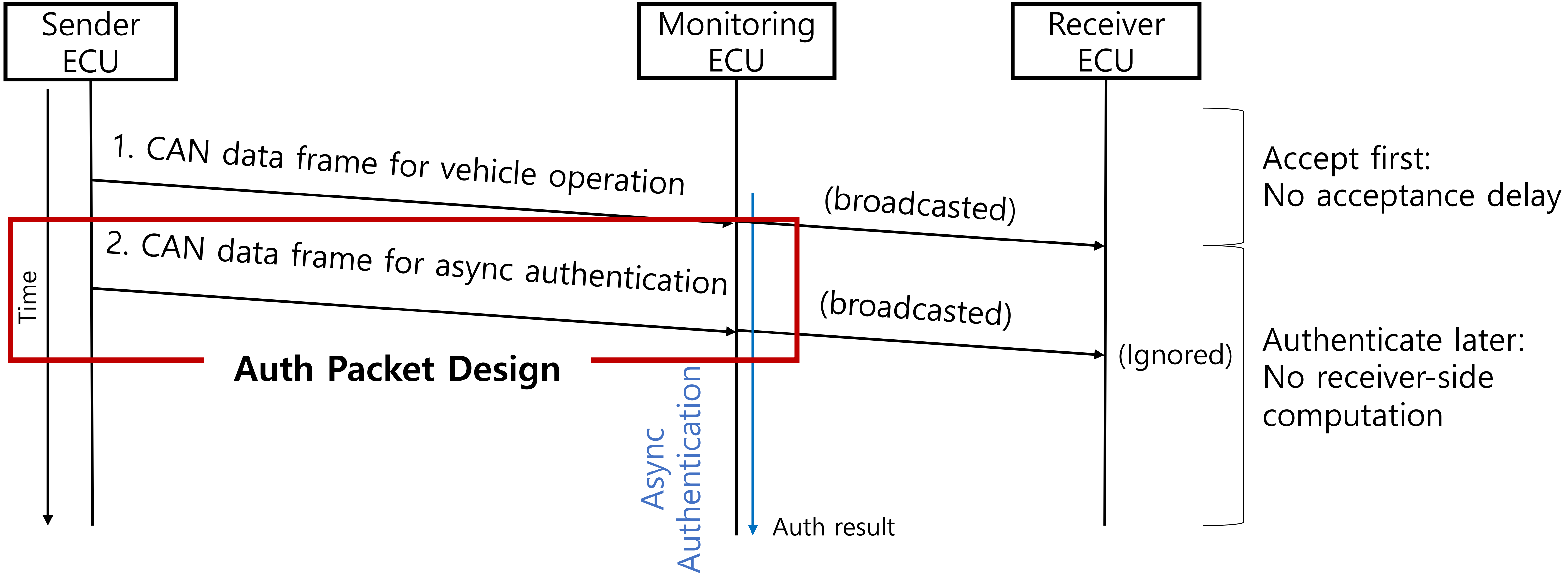
ShadowAuth Design

"Accept-first-authenticate-later"



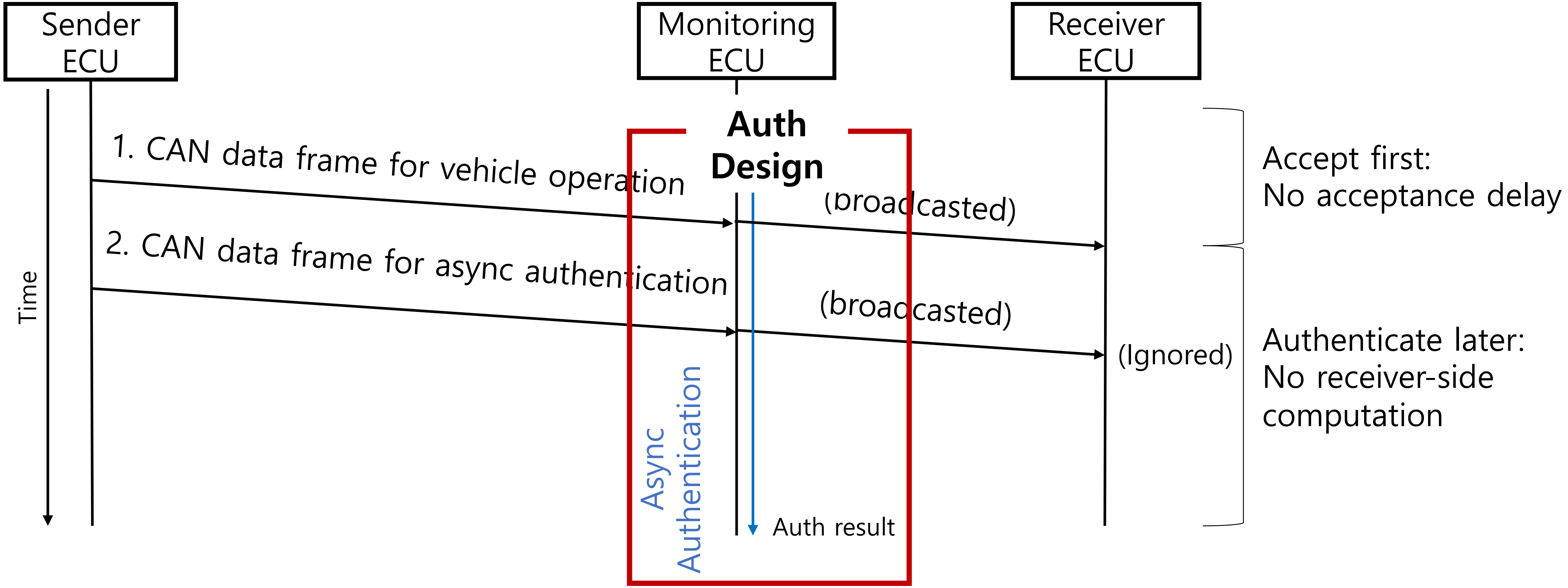
ShadowAuth Design

"Accept-first-authenticate-later"



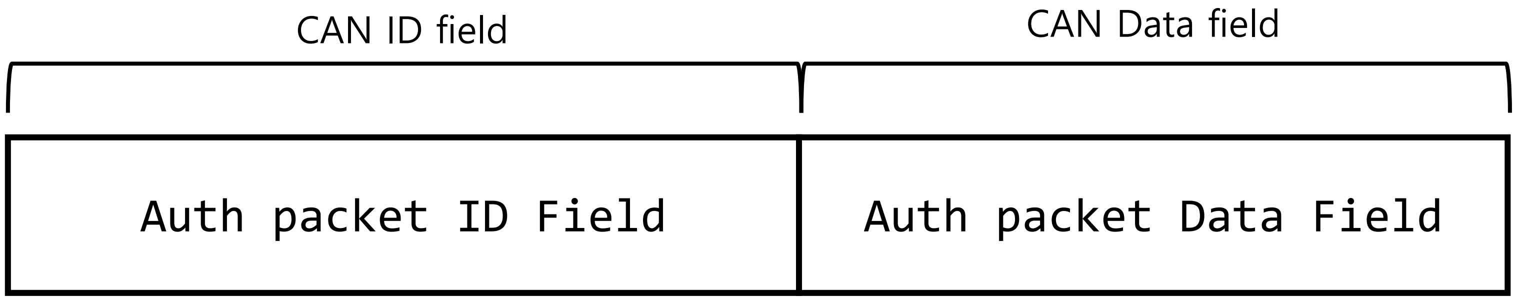
ShadowAuth Design

"Accept-first-authenticate-later"



Authentication Packet Design

- Design goal
  - Compatible with existing protocols
  - Minimize impact on existing systems
  - Tolerate known attacks



CAN ID field

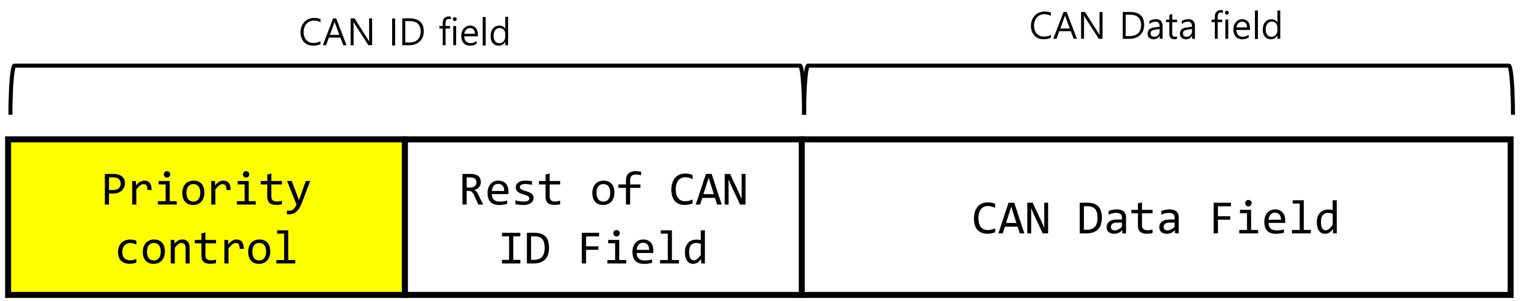
CAN Data field

Auth packet ID Field

Auth packet Data Field

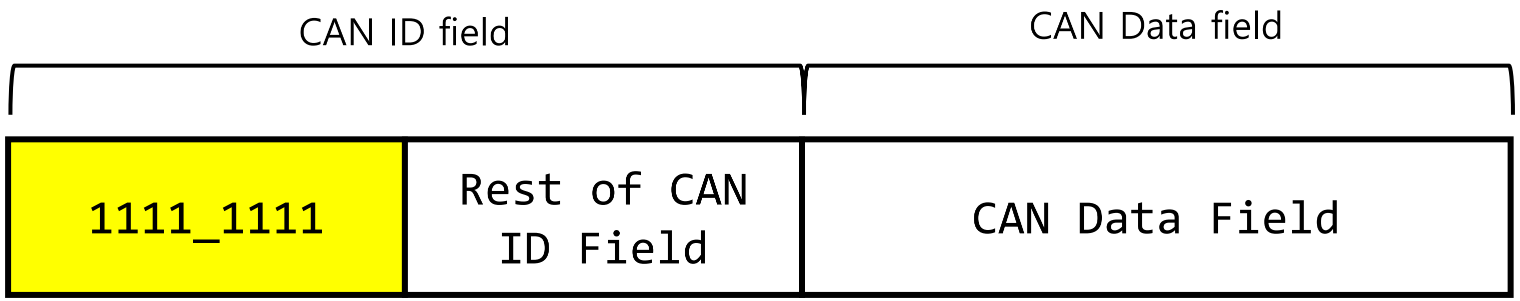
Authentication Packet Design

- Priority Control Field: Variable Length of Sequential Recessive Bits (1)
  - Make packets compatible with existing protocols
  - Minimize impact on existing systems: always yield the bus to op packets



Authentication Packet Design

- Concrete Example: J1939 Standard
  - Authentication packets start with sequential eight recessive bits
  - No J1939 packet starts with those eight



CAN ID field

CAN Data field

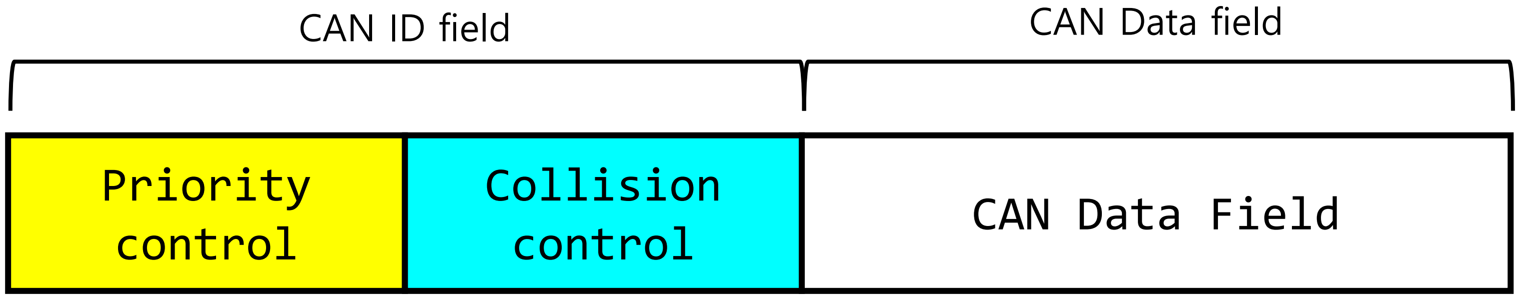
1111_1111

Rest of CAN ID Field

CAN Data Field

Authentication Packet Design

- Collision Control Field: Randomized Field
  - Avoid collision among auth packets
  - Hide auth packets from attackers
  - Tolerate to op-auth pair recovery attacks



CAN ID field

CAN Data field

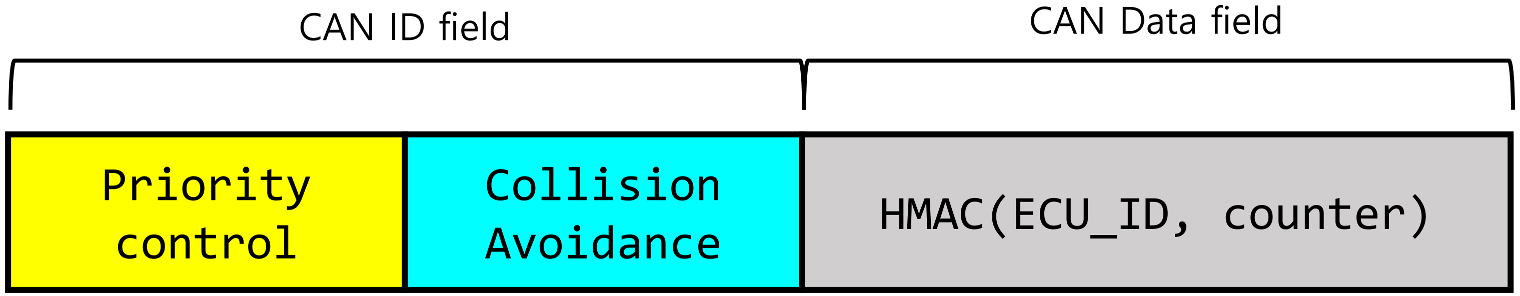
Priority control

Collision control

CAN Data Field

Authentication Packet Design

- Data Field: HMAC (Blake3)
  - Sender authentication with a unique ECU ID
  - Tolerate to replay attacks: increasing counter
  - Minimize potential impact on existing systems
    - Minimize firmware patching: no access to firmware-side data (e.g., CAN ID)
    - Minimize traffic increase by using a single packet for auth



CAN ID field

CAN Data field

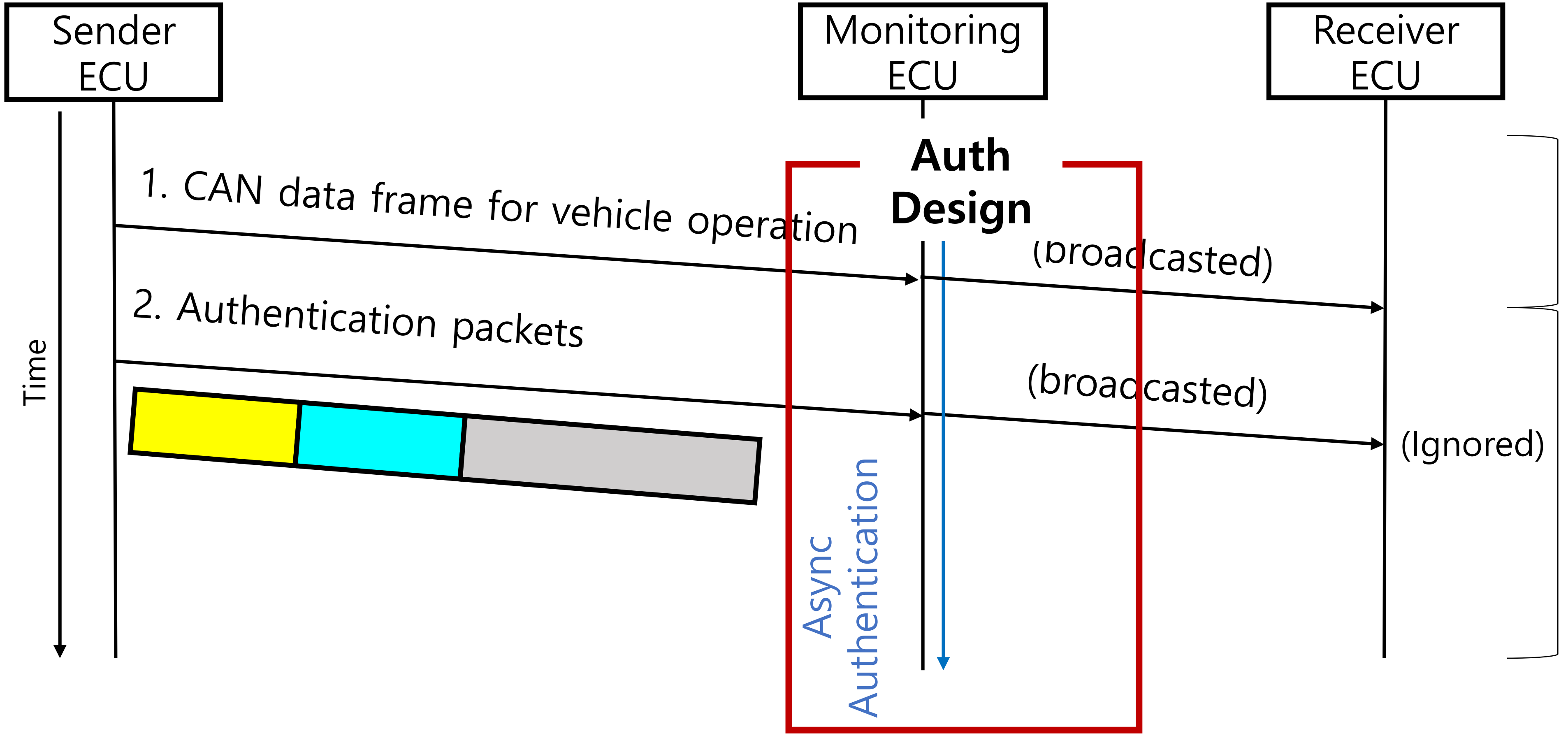
Priority control

Collision Avoidance

HMAC(ECU_ID, counter)

ShadowAuth Design

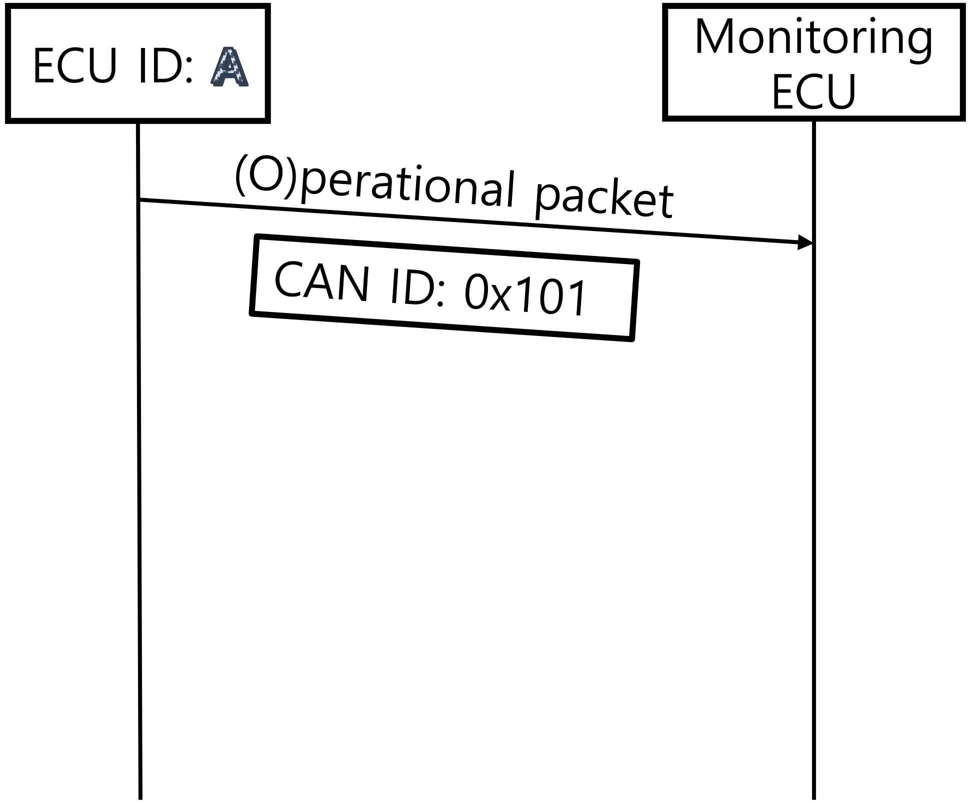
"Accept-first-authenticate-later"



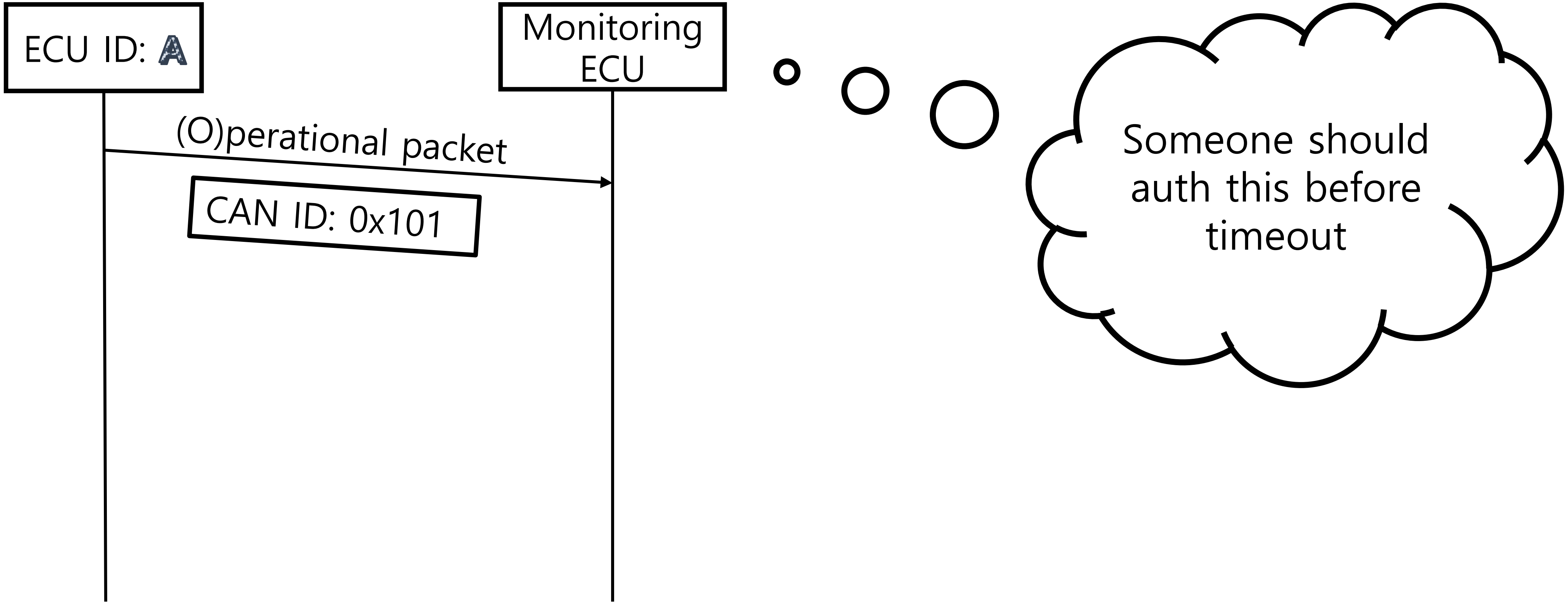
Accept first:
No acceptance delay

Authenticate later:
No receiver-side computation

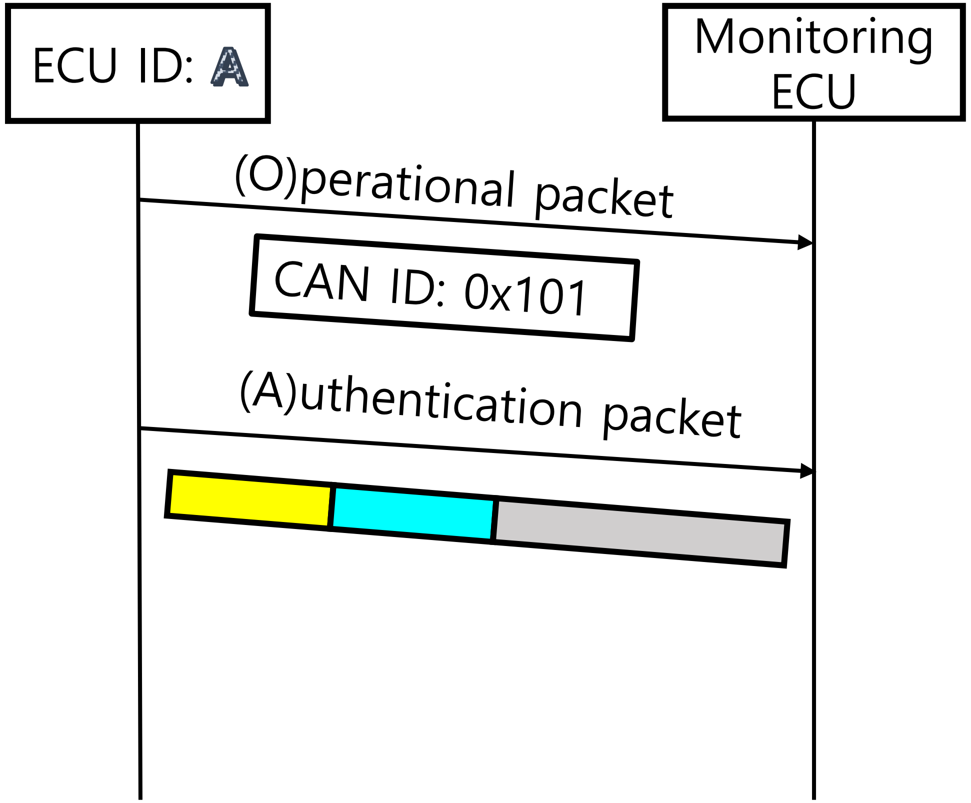
Authentication Design



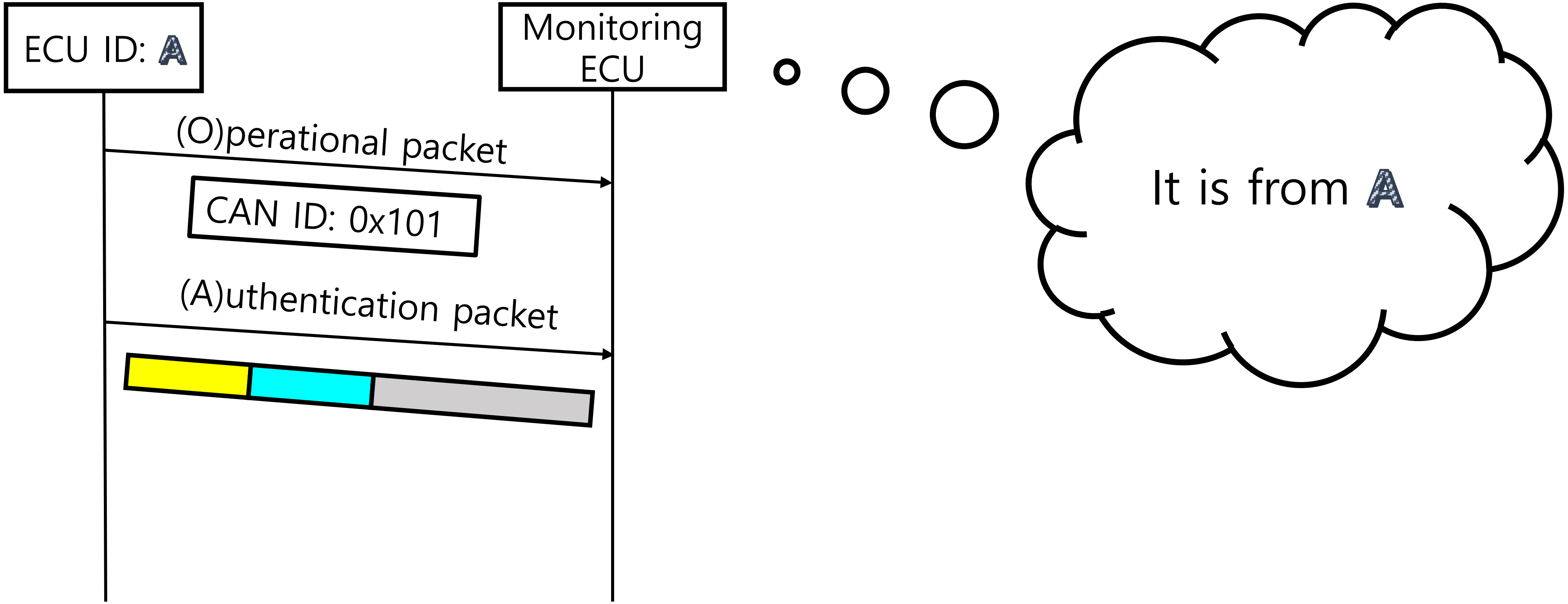
Authentication Design



Authentication Design



Authentication Design



ECU ID: A

Monitoring ECU

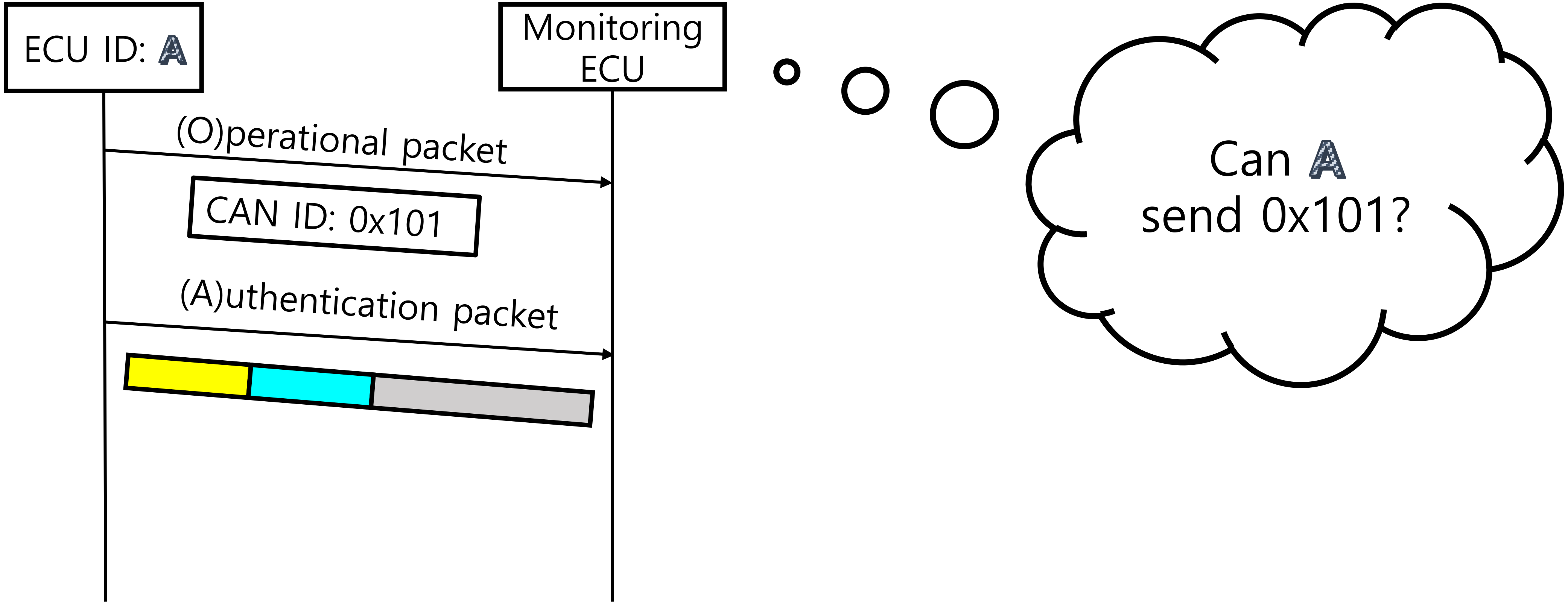
(O)perational packet

CAN ID: 0x101

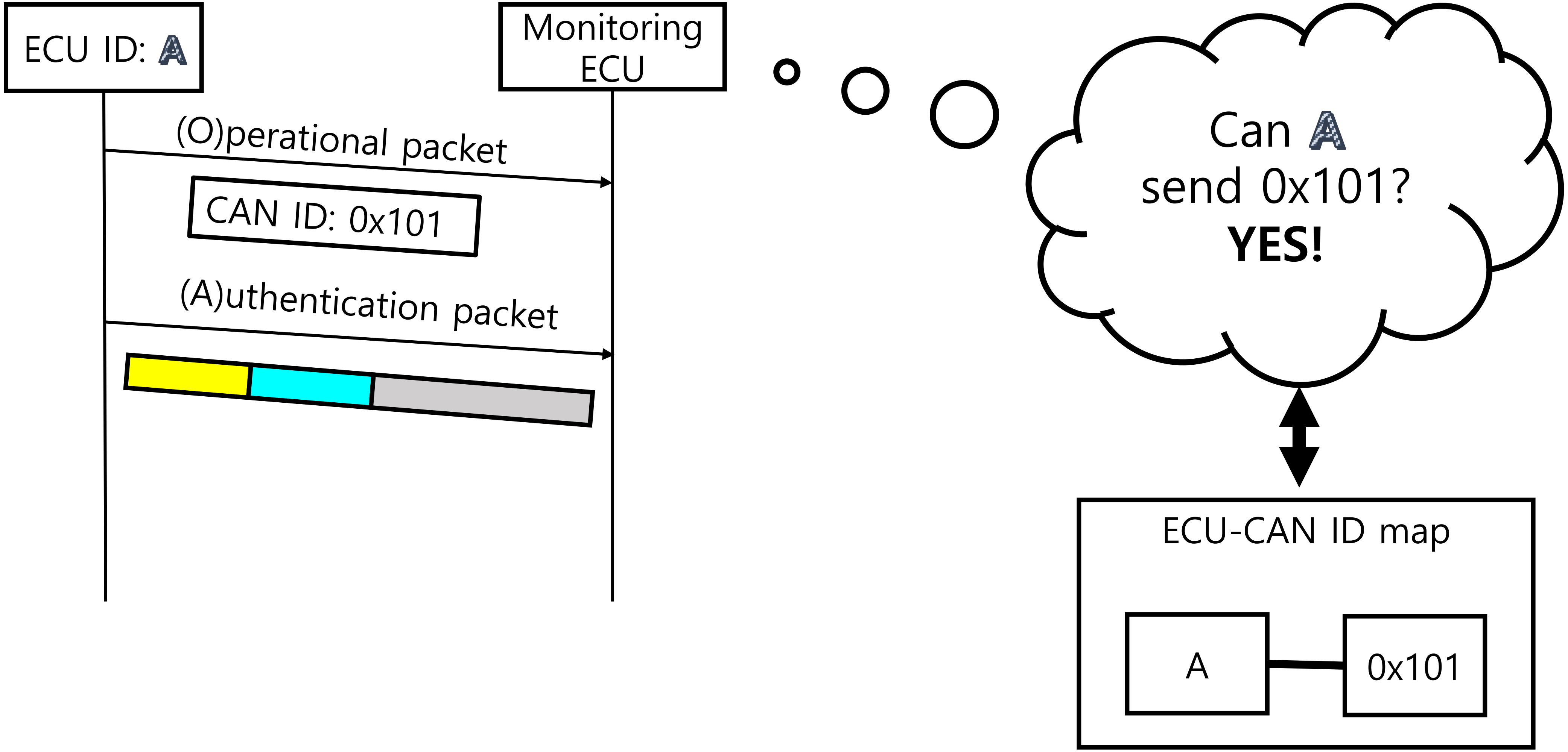
(A)uthentication packet

It is from A

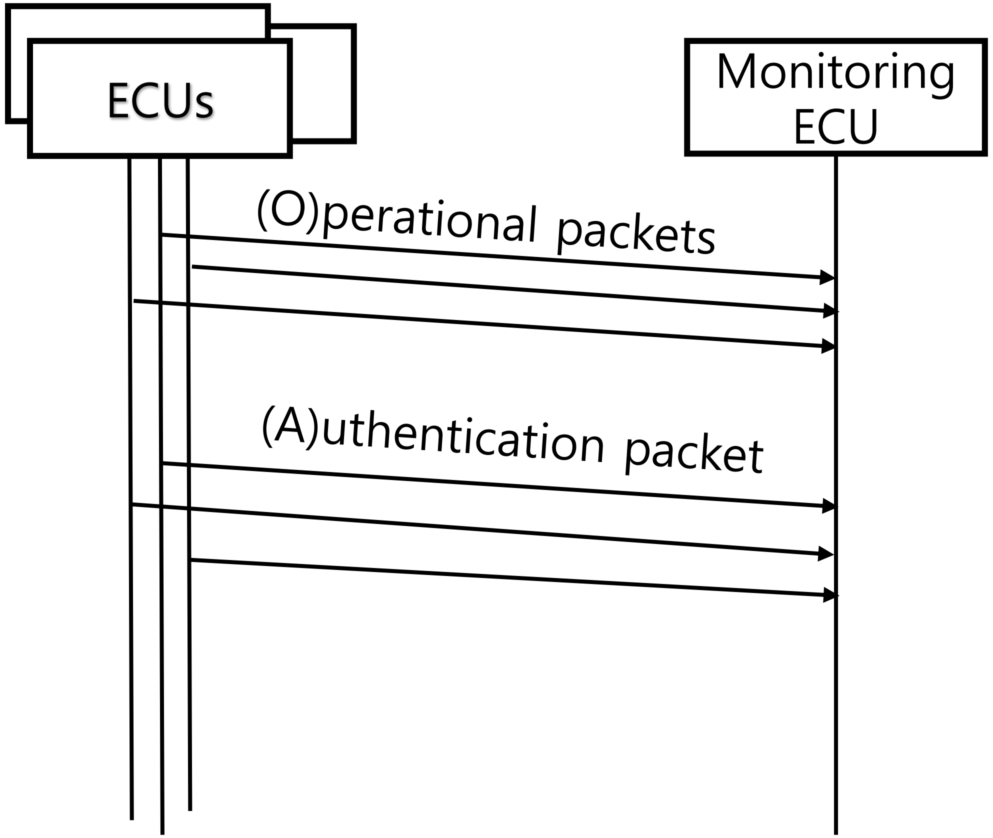
Authentication Design

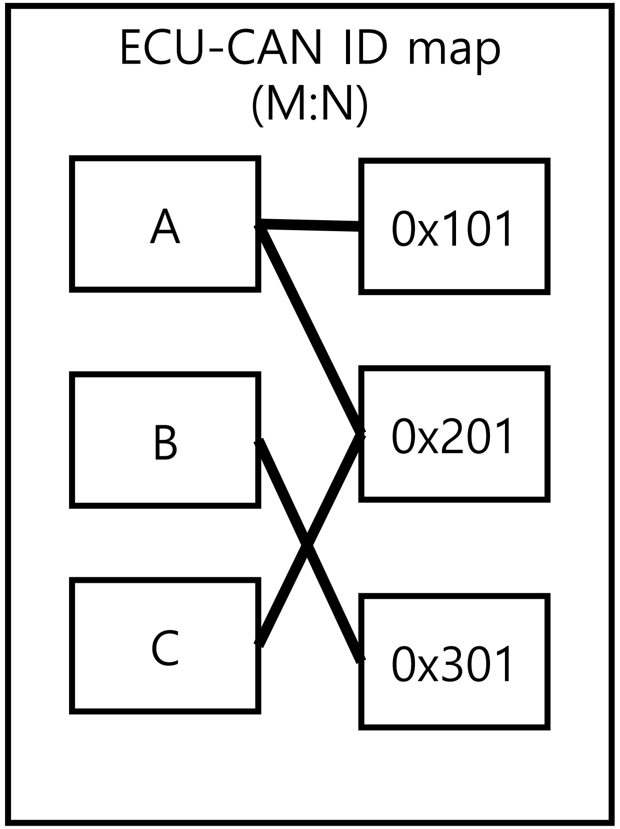


Authentication Design

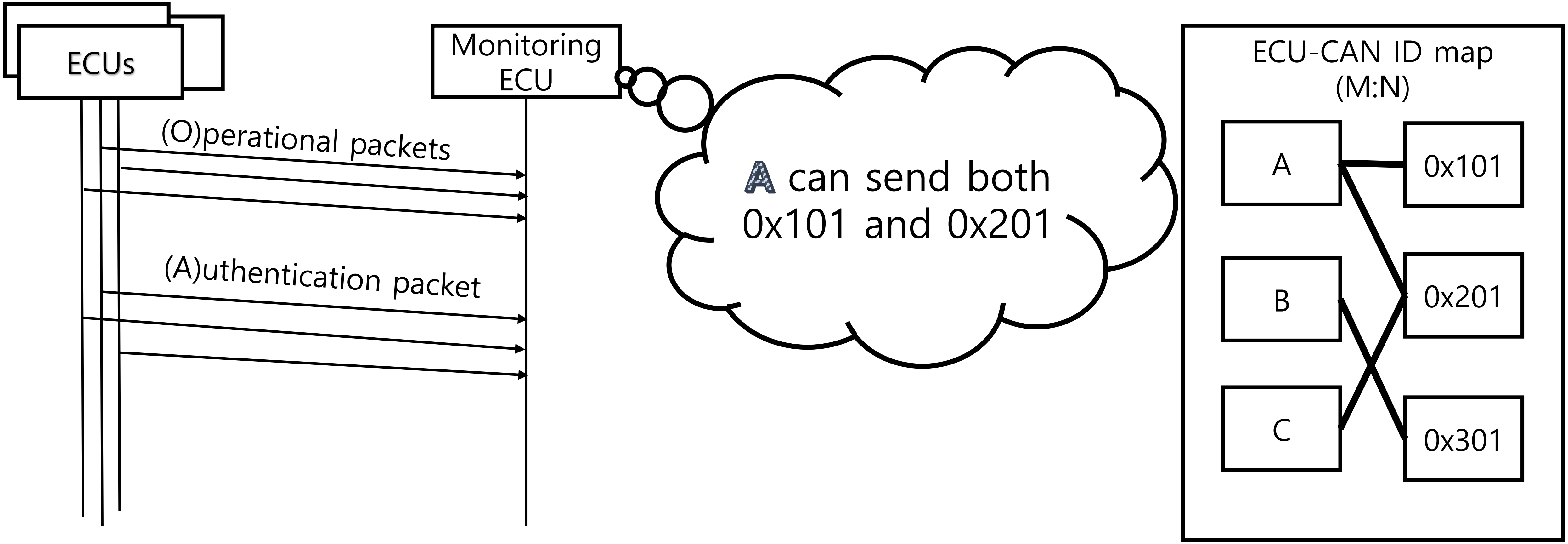


Authentication Design

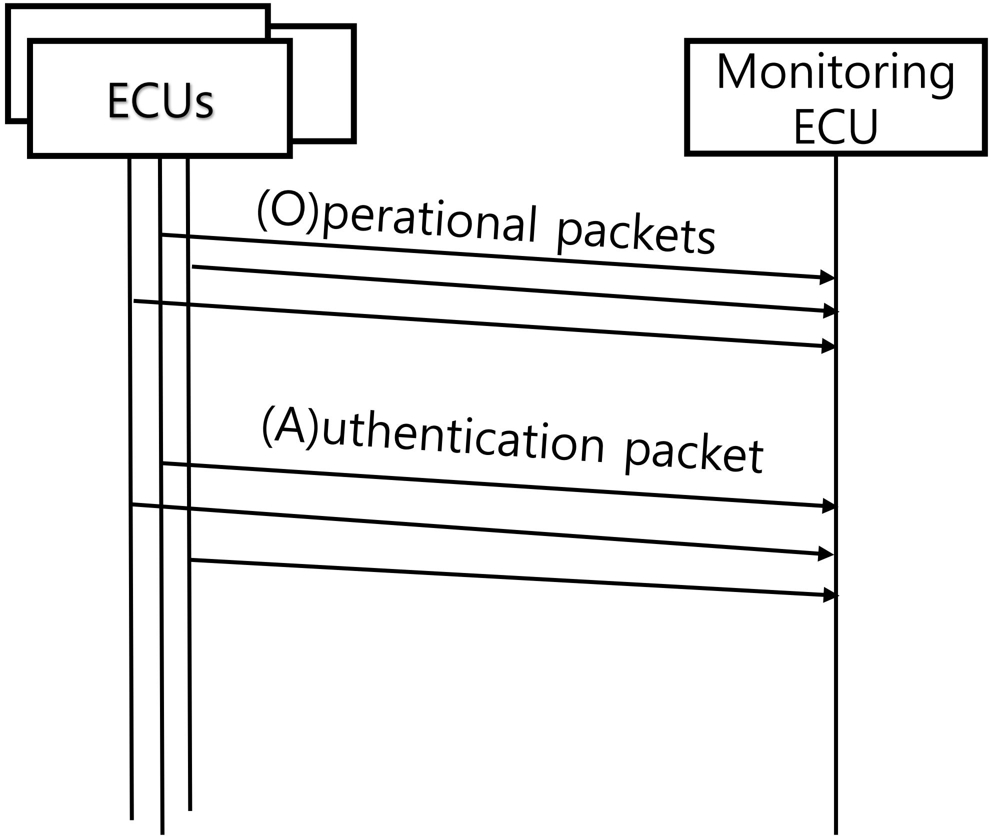




Authentication Design



Authentication Design



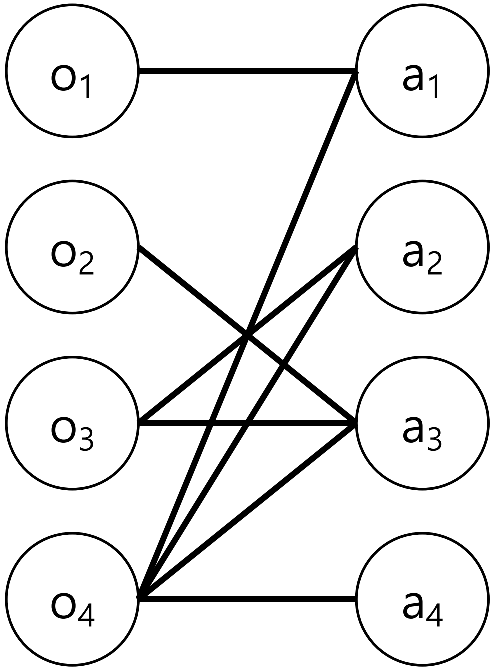
ECUs

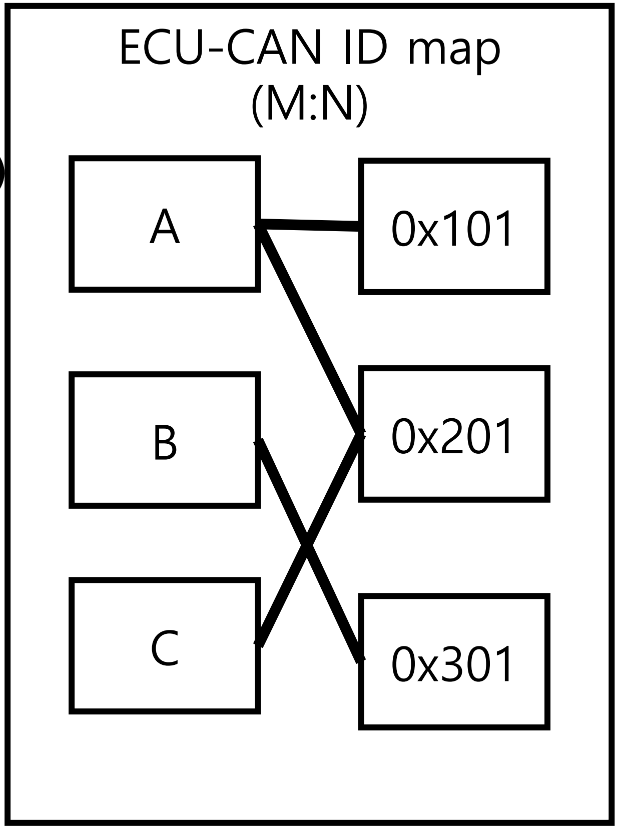
Monitoring ECU

(O)perational packets

(A)uthentication packet

Too many possible scenarios





ECU-CAN ID map (M:N)

A

B

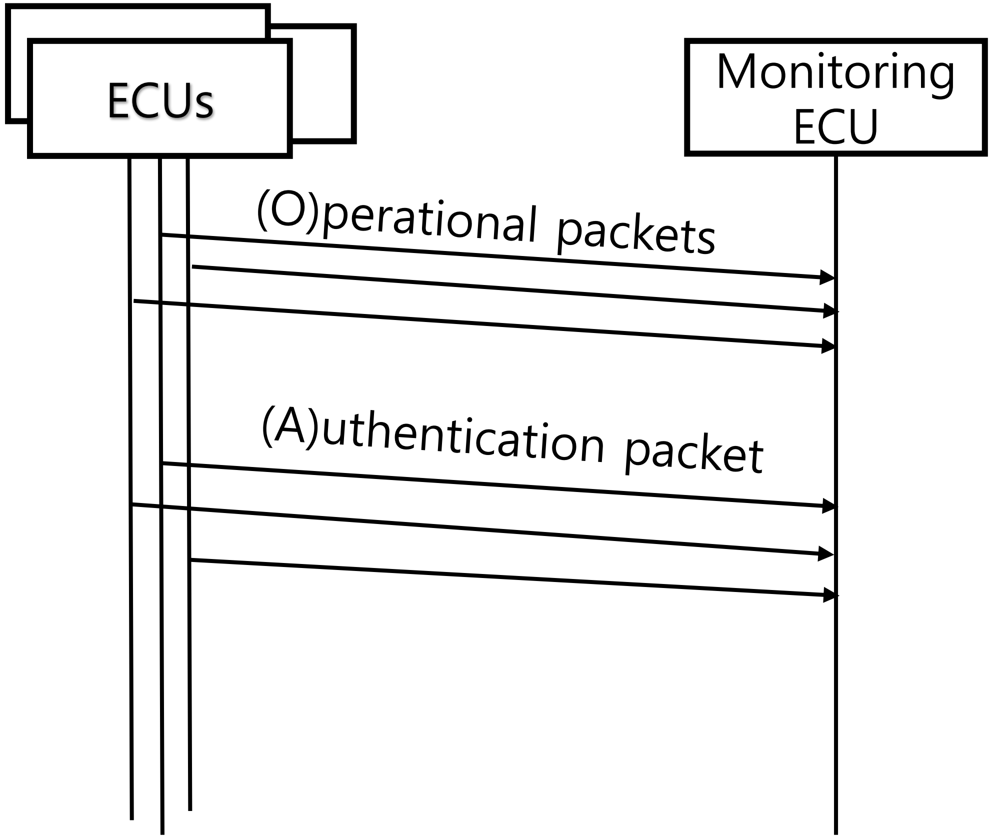
C

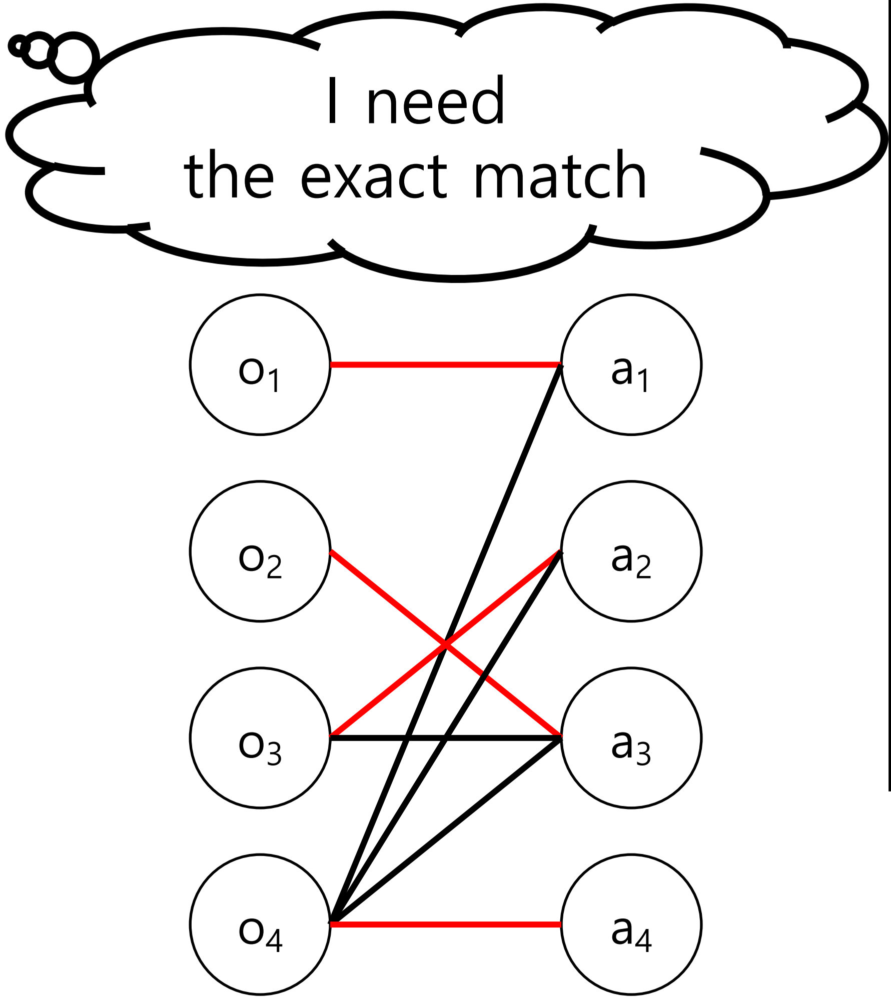
0x101

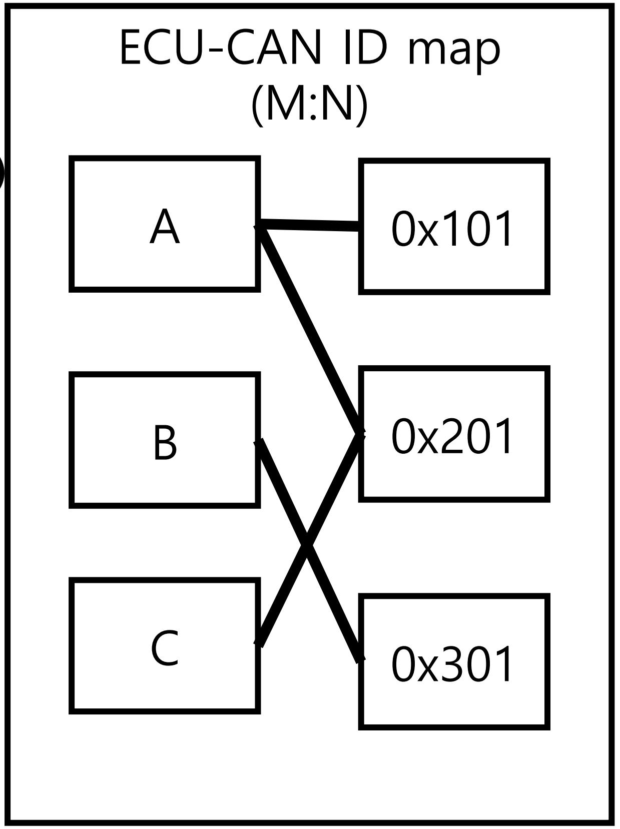
0x201

0x301

Authentication Design

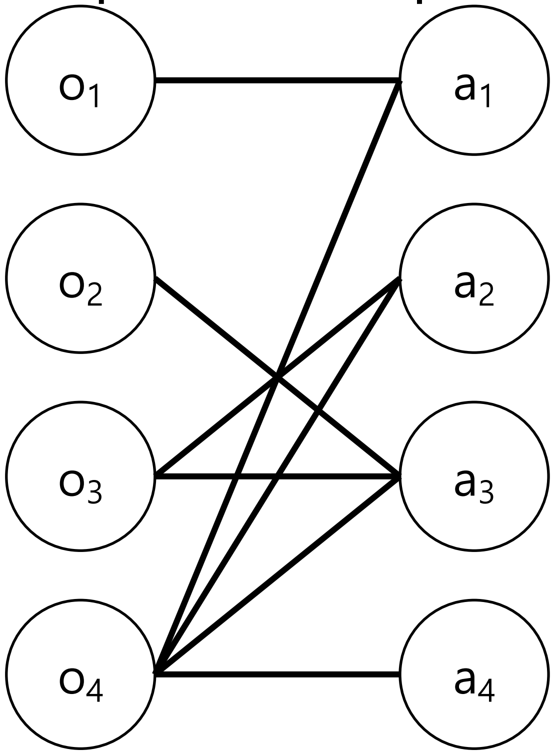




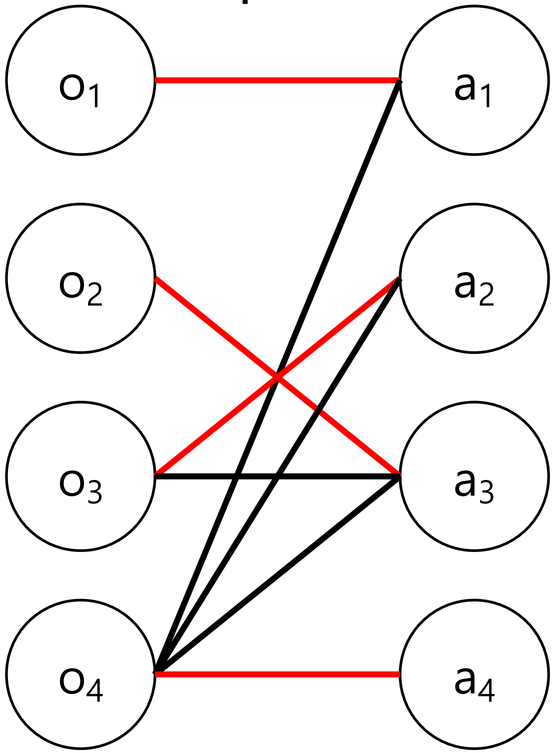


Authentication Design

- Observation: maximum matching should be equal to the number of auth packets
    - If no attack presents, operational and authentication packets are sent once



Bipartite graph

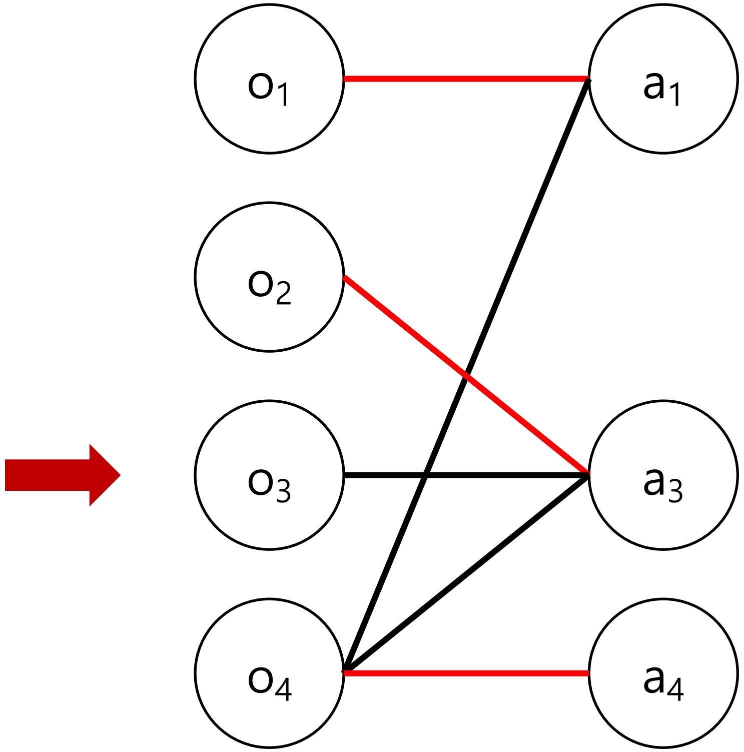


Maximum matching (red)

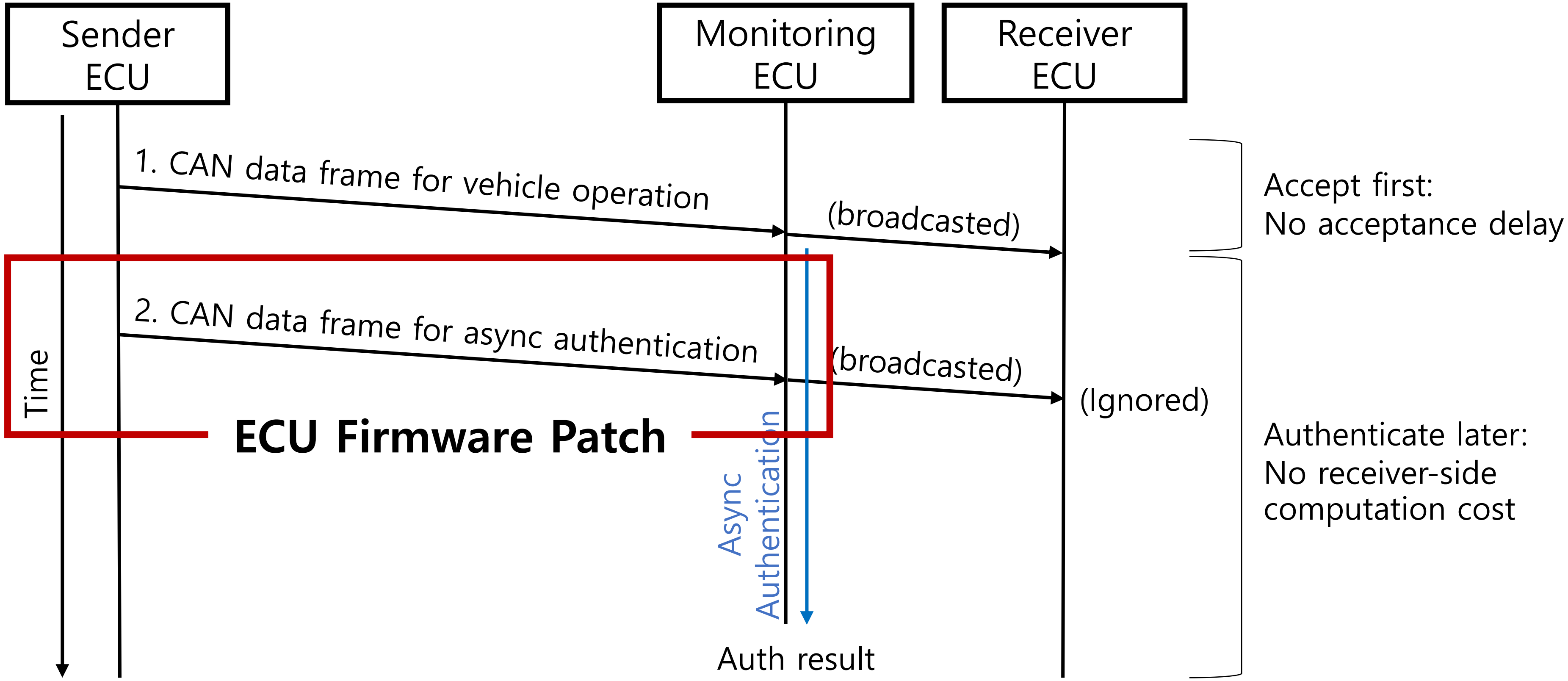
Authentication Design

- Attack scenario: Operational packet fabrication

O_3 does not match with any of auth packet!
We are under attacks!



ShadowAuth Design

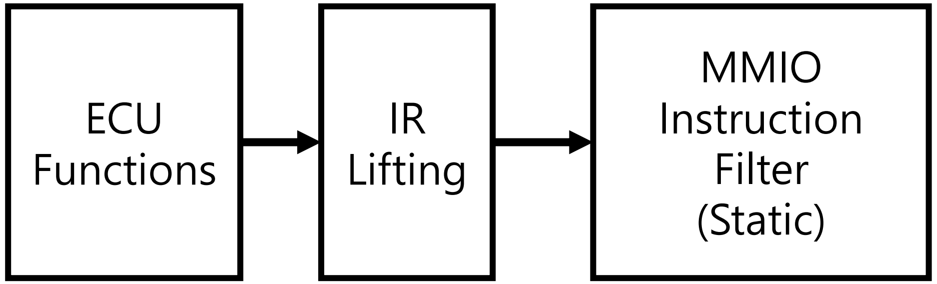


- Where: Every instruction next to `call can_tx`
- How: trampoline-based patching

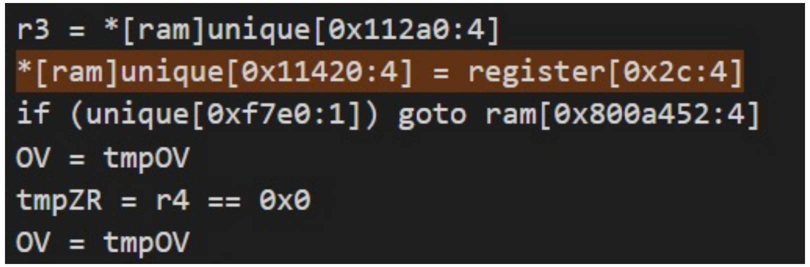
ECU Firmware Patch

- Goal:
  - Automation for finding the CAN Tx function
  - Independent to architecture
- Solution:
  - Check the MMIO instruction’s volume

ECU Firmware Patch

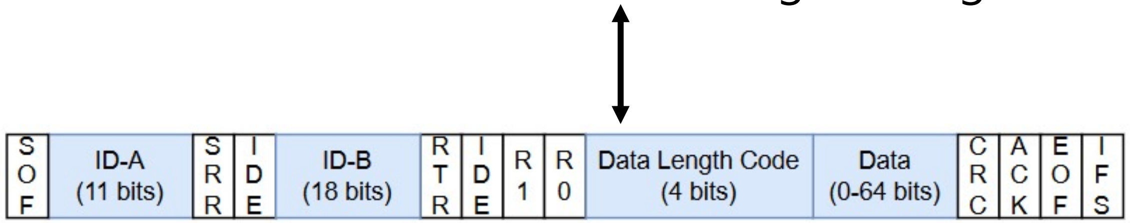


- Static analysis



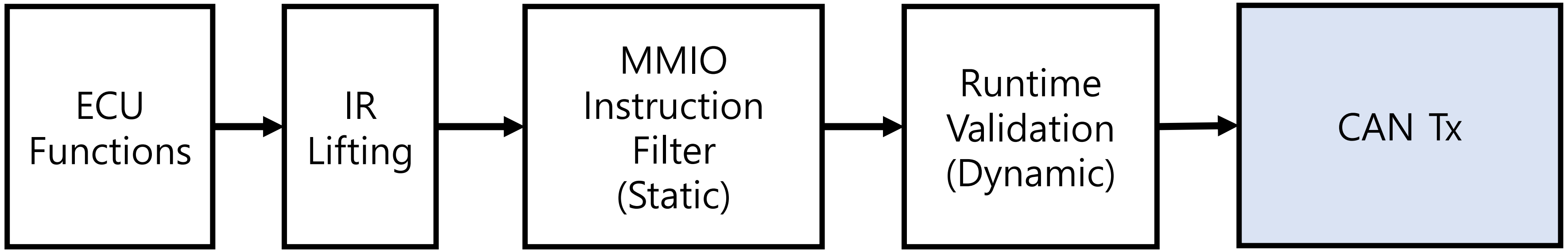
```
r3 = *[ram]unique[0x112a0:4]
*[ram]unique[0x11420:4] = register[0x2c:4]
if (unique[0xf7e0:1]) goto ram[0x800a452:4]
OV = tmpOV
tmpZR = r4 == 0x0
OV = tmpOV
```

MMIO instructions are large enough to send CAN packet?

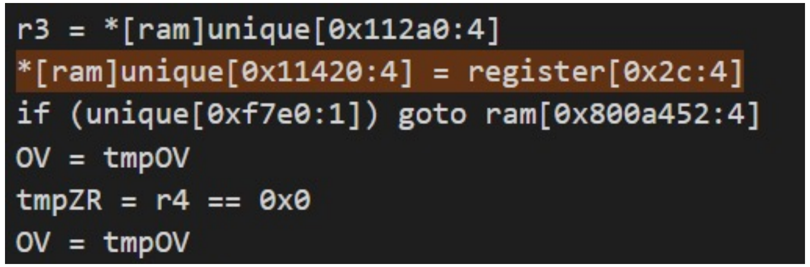


SOF	ID-A (11 bits)	SRR	IDE	ID-B (18 bits)	RTR	IDE	R1	R0	Data Length Code (4 bits)	Data (0-64 bits)	CRC	ACK	EOF	IFS

ECU Firmware Patch


ECU Functions
IR Lifting
MMIO Instruction Filter (Static)
Runtime Validation (Dynamic)
CAN Tx

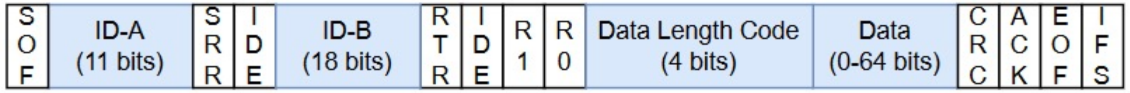
- Static analysis



```
r3 = *[ram]unique[0x112a0:4]
*[ram]unique[0x11420:4] = register[0x2c:4]
if (unique[0xf7e0:1]) goto ram[0x800a452:4]
OV = tmpOV
tmpZR = r4 == 0x0
OV = tmpOV
```

- Dynamic Analysis
  - Monitor the actual CAN bus
  - Run functions

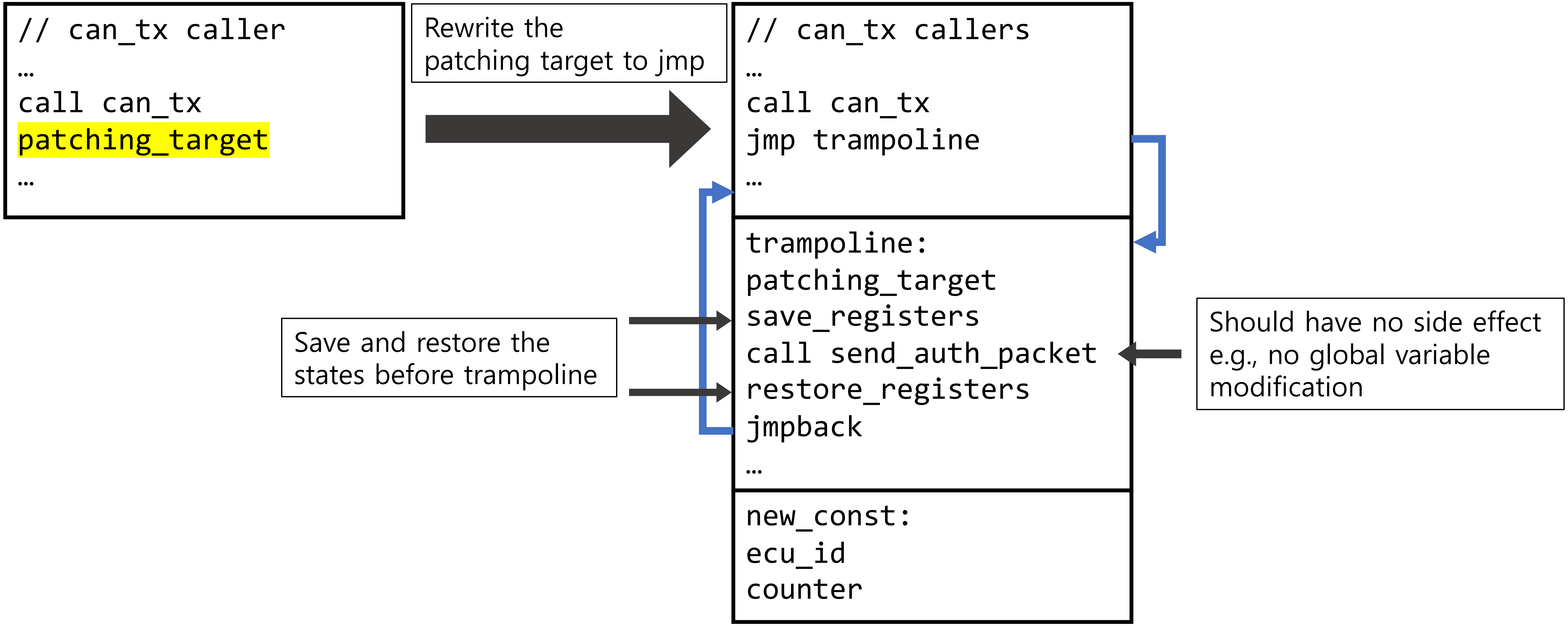
MMIO instructions are large enough to send CAN packet?



SOF	ID-A (11 bits)	SRR	IDE	ID-B (18 bits)	RTR	IDE	R1	R0	Data Length Code (4 bits)	Data (0-64 bits)	CRC	ACK	EOF	IFS

ECU Firmware Patch

Trampoline-based patch



```
// can_tx caller
…
call can_tx
patching_target
…
```

Rewrite the patching target to jmp

```
// can_tx callers
…
call can_tx
jmp trampoline
…
```

```
trampoline:
patching_target
save_registers
call send_auth_packet
restore_registers
jmpback
…
```

```
new_const:
ecu_id
counter
```

Save and restore the states before trampoline

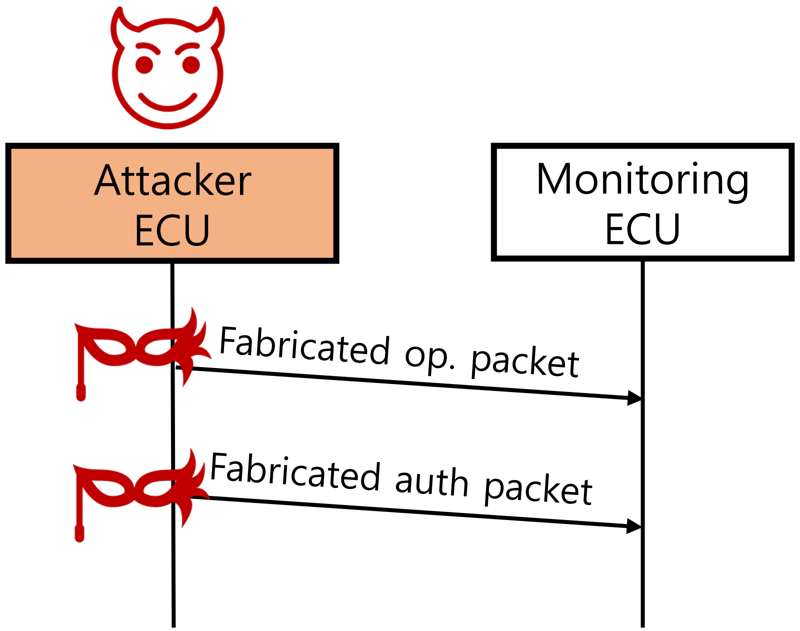
Should have no side effect e.g., no global variable modification

Evaluation

- Open-source ECU firmware: rusEFI, Rabbit ECU, Styreehet
- Real-world CAN traffic: 2014 Kenworth T270 / 2015 Kenworth T660 (>37M packets)
- Static analysis: 100% accuracy (confirmed by dynamic analysis)
- Asynchronous authentication: 14ms (worst)
- Space overhead (~~182KB~~ 84KB by Blake3)

Evaluation: Case Study

- Case 1: Impersonation attack
  - Auth fails: ECU_ID is unknown

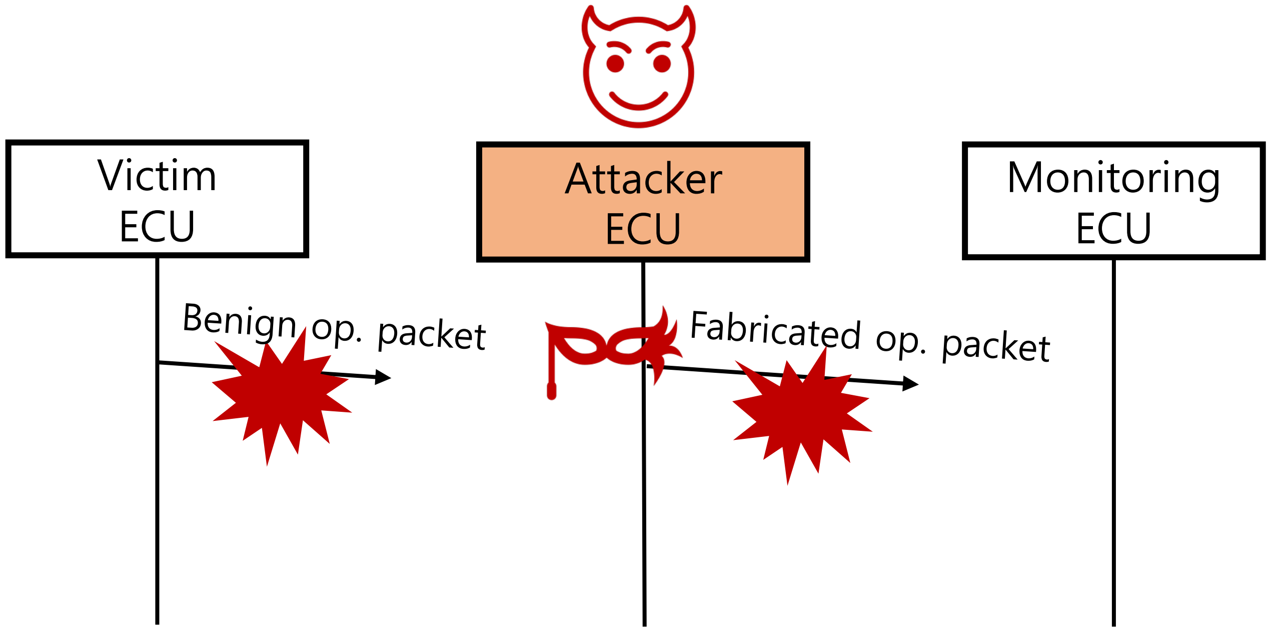


Priority control	Anonymity control	HMAC(ECU_ID, counter)

Authentication packet

Evaluation: Case Study

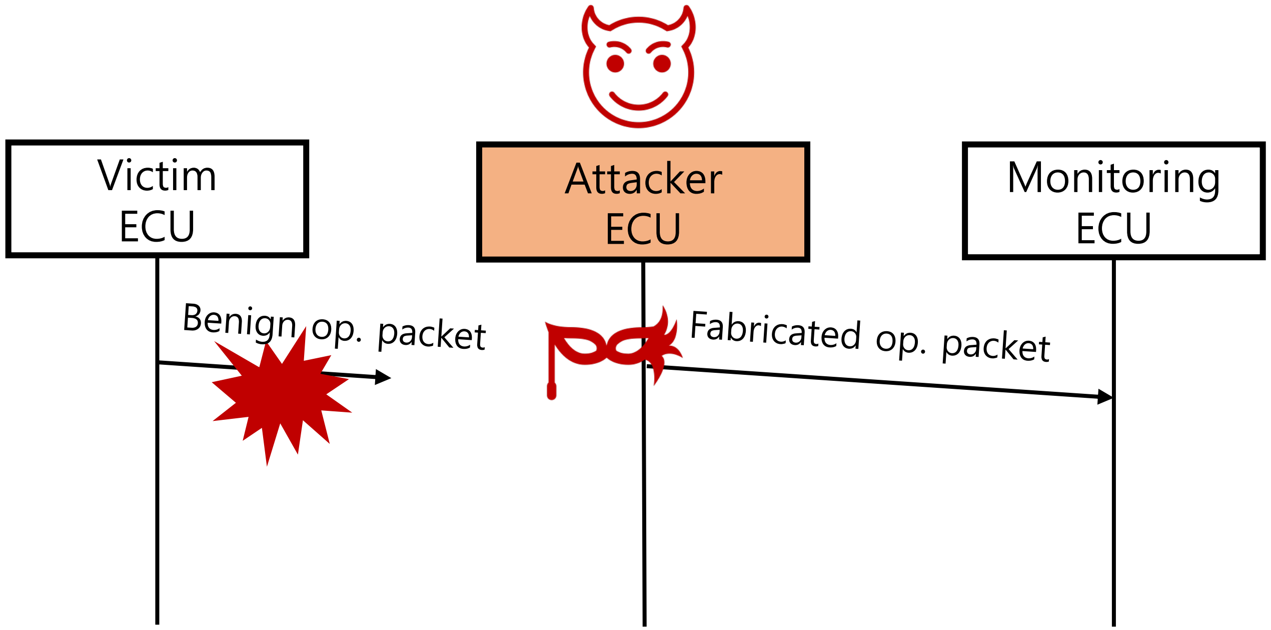
- Case 2: Bus-off attack



Make collision deliberately

Evaluation: Case Study

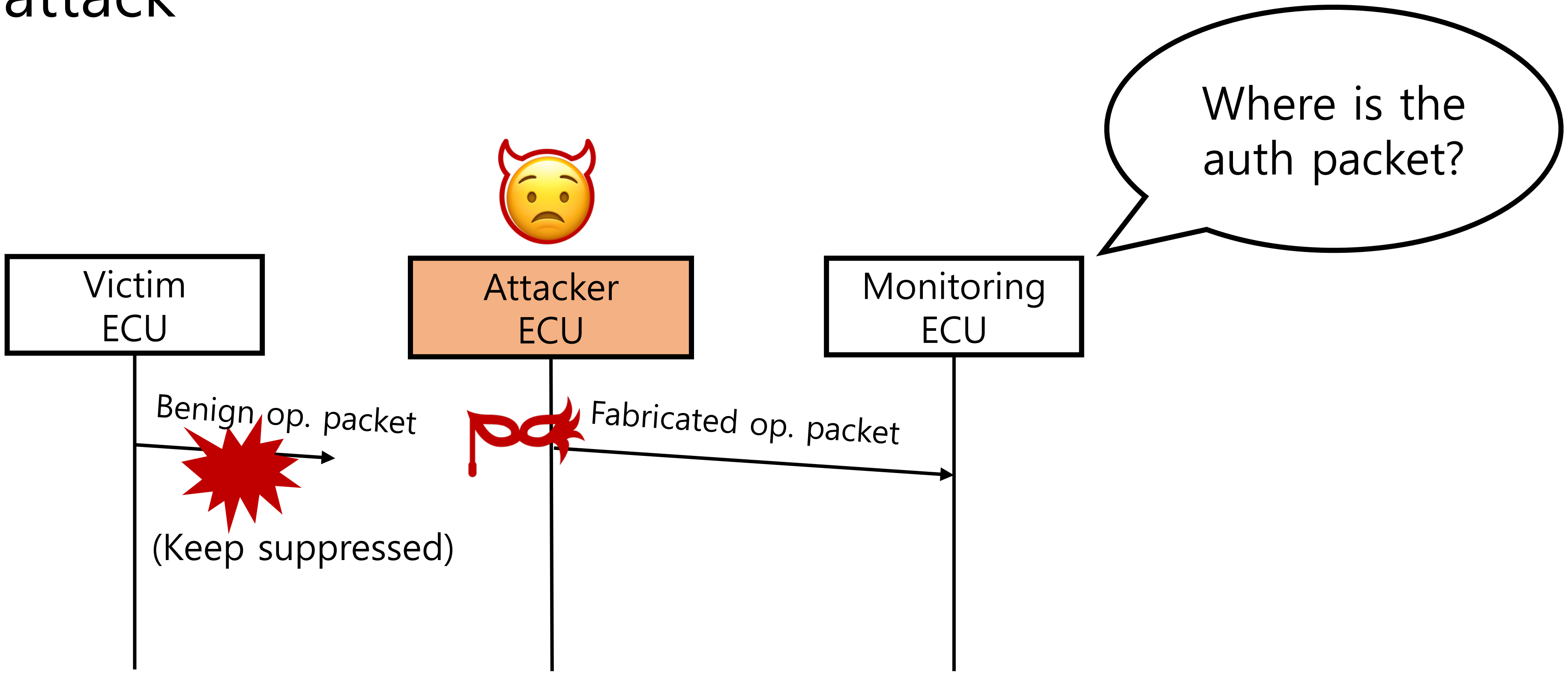
- Case 2: Bus-off attack



* The victim ECU turns into the error-passive mode first

Evaluation: Case Study

- Case 2: Bus-off attack



Limitations

- Potential impact of “accept-first”
- Recovery attacks on ECU ID, ECU-CAN ID mapping, and counter
    - E.g., firmware dump through OBD II
- Truncated Blake3 to 64-bit
- Patching real-world ECU firmware in the user side

Conclusion

- Vehicles are increasingly exposed to remote attacks
- Easily deployable solutions are required
- ShadowAuth proposes:
  - Backward-compatible CAN authentication scheme
  - Automated patch with trampoline-based binary rewriting
  - Real-time authentication with accept-first-authenticate-later policy

- Feel free to download: github.com/purseclab/ShadowAuth
  - Static analysis
  - ECU firmware patching

Thank you!

sk@purdue.edu

github.com/purseclab/ShadowAuth

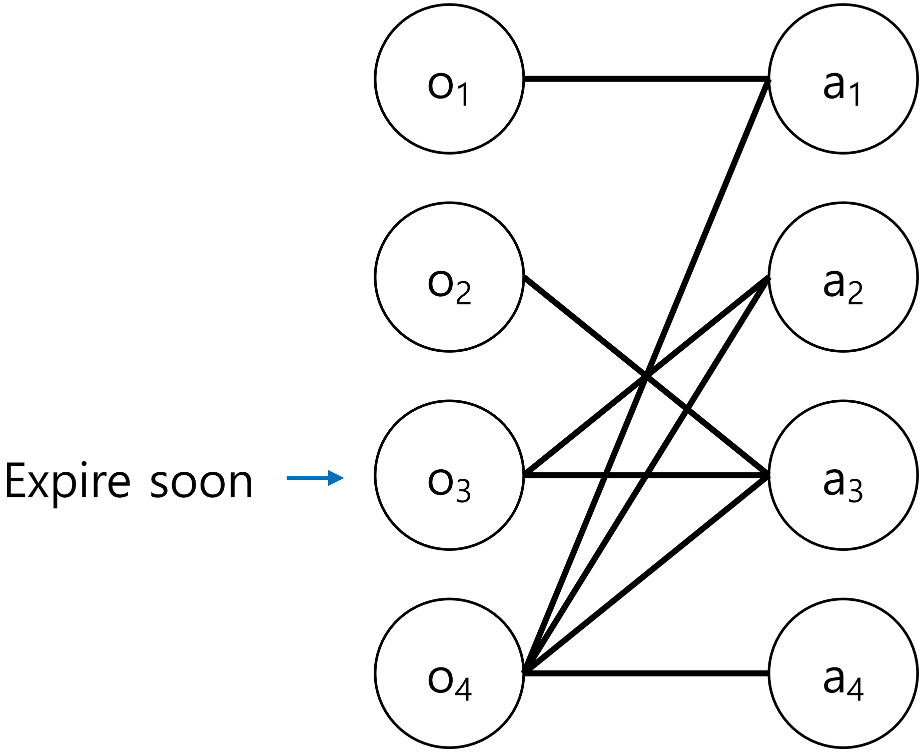
PURDUE UNIVERSITY

PurSec Lab

This work is supported in part by ONR Grant #N00014-18-1-2674 and #N00014-20-1-2128, NSF Award #CNS-2145744, DARPA contract #N6600120C4031, and NRF Grant #2021R1F1A1049822

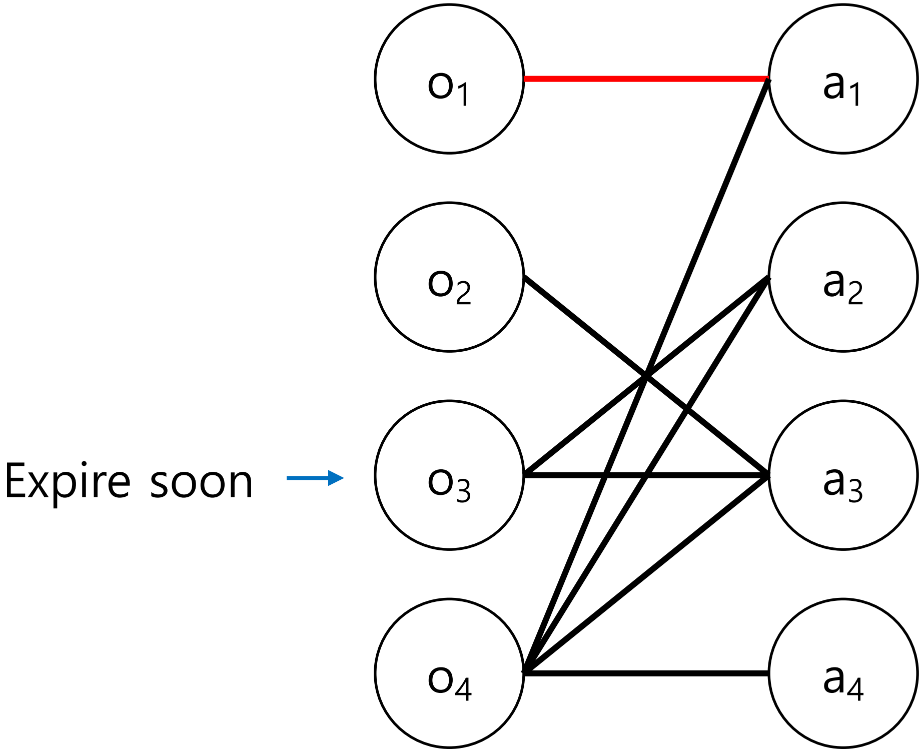
Authentication Design

- Match vertices: fewer edges first



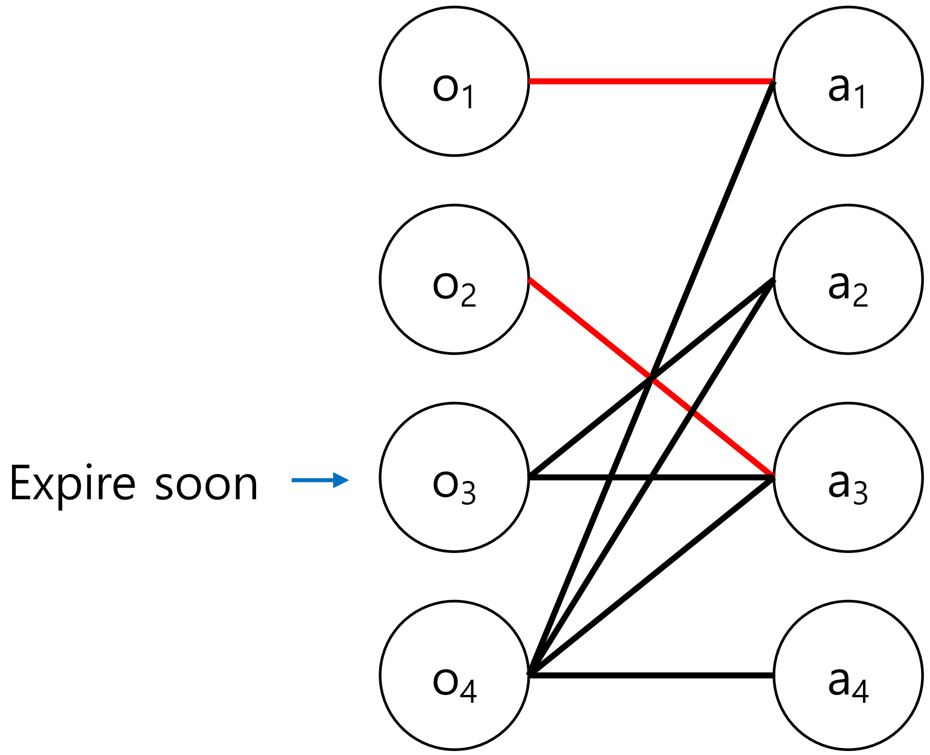
Authentication Design

- Match vertices: fewer edges first



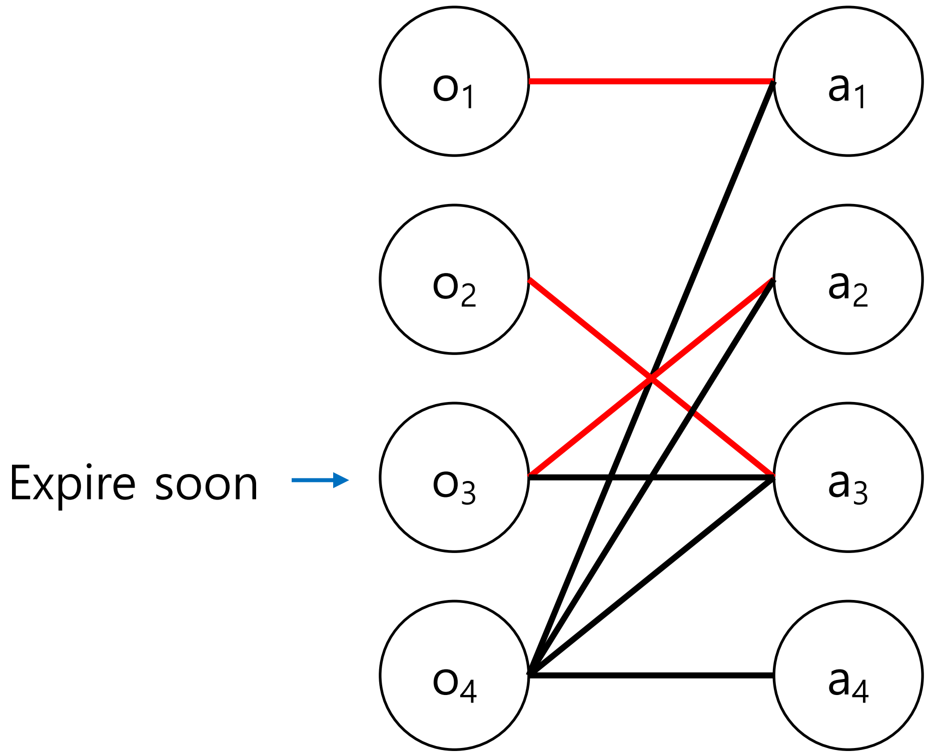
Authentication Design

- Match vertices: fewer edges first



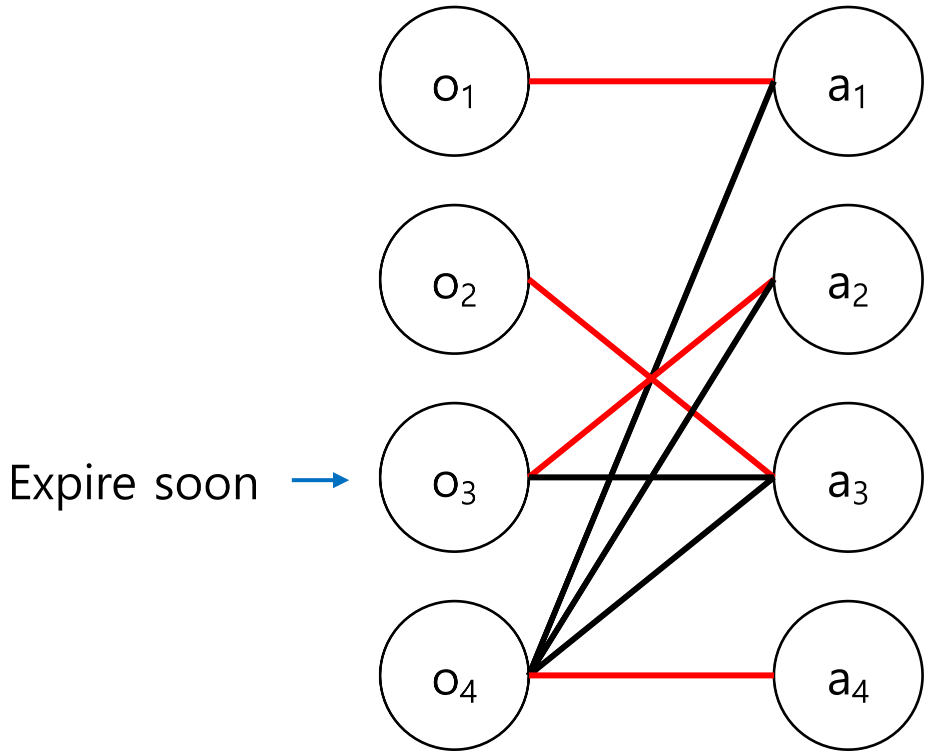
Authentication Design

- Match vertices: fewer edges first



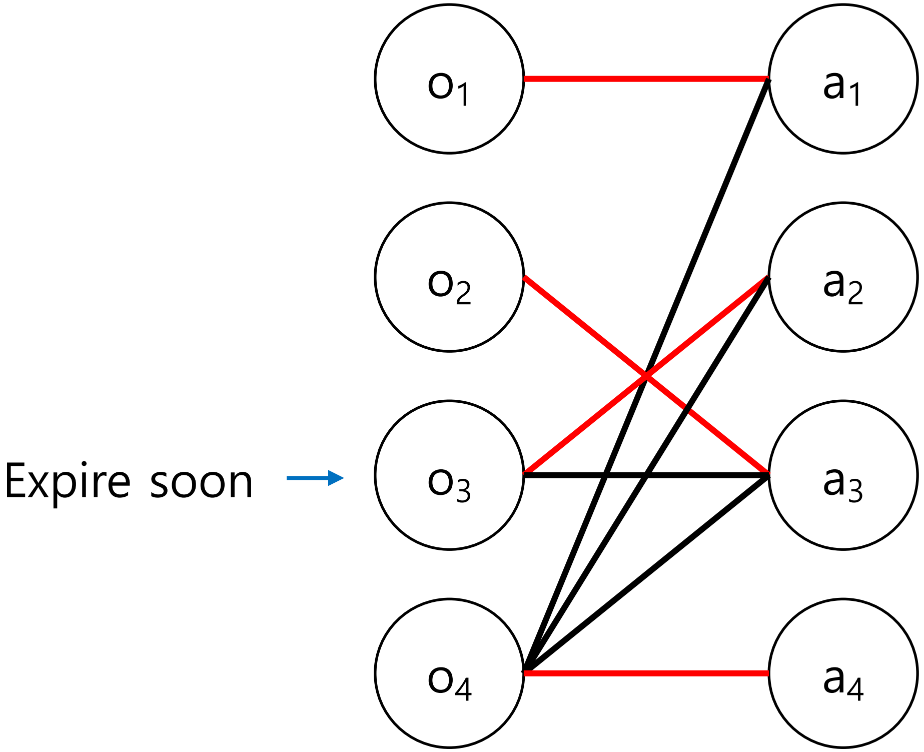
Authentication Design

- Match vertices: fewer edges first



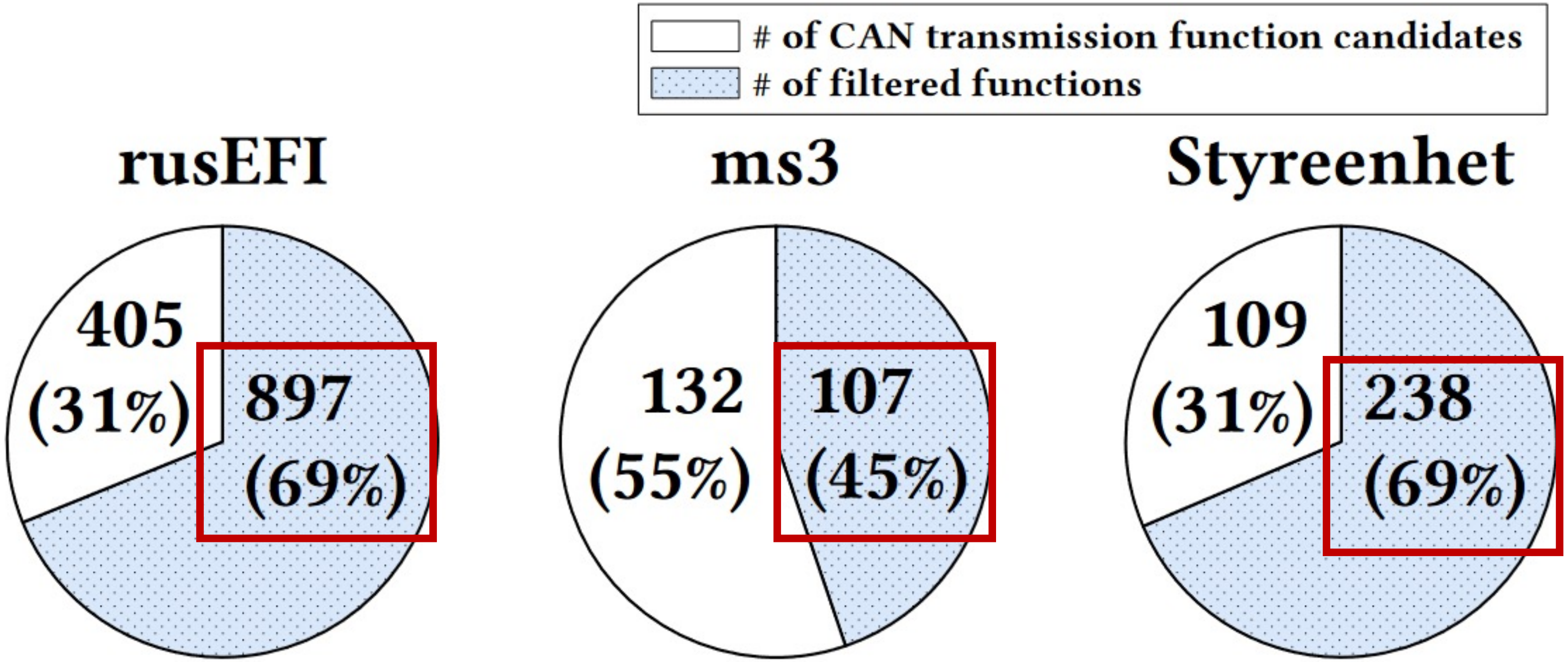
Authentication Design

- Match vertices: fewer edges first

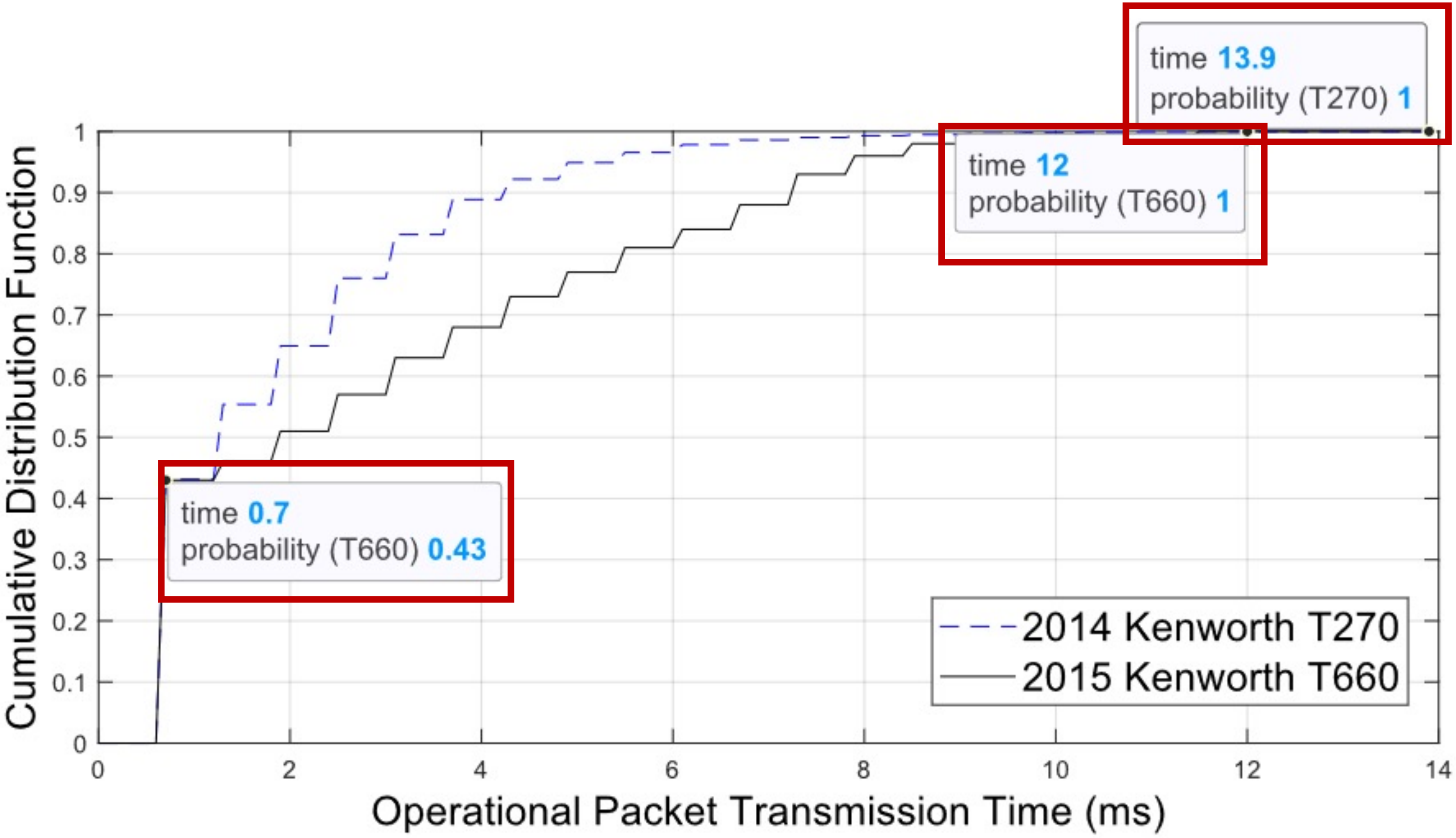


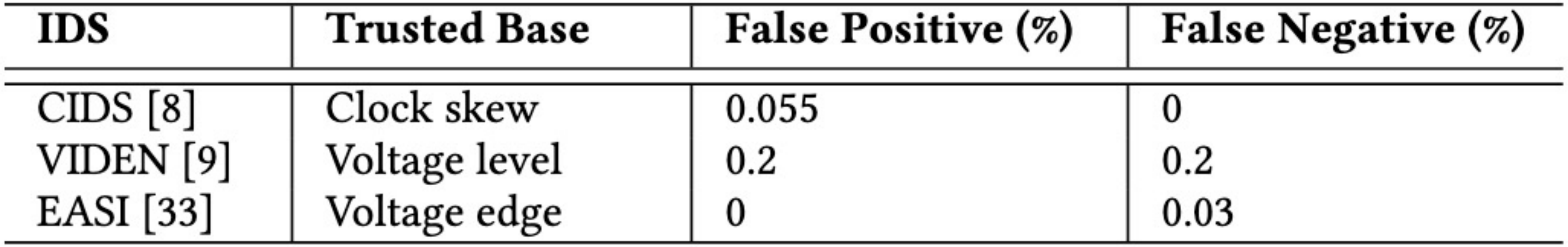
Match = # of auth packets
Authentic!

Evaluation: Static Analysis Efficiency



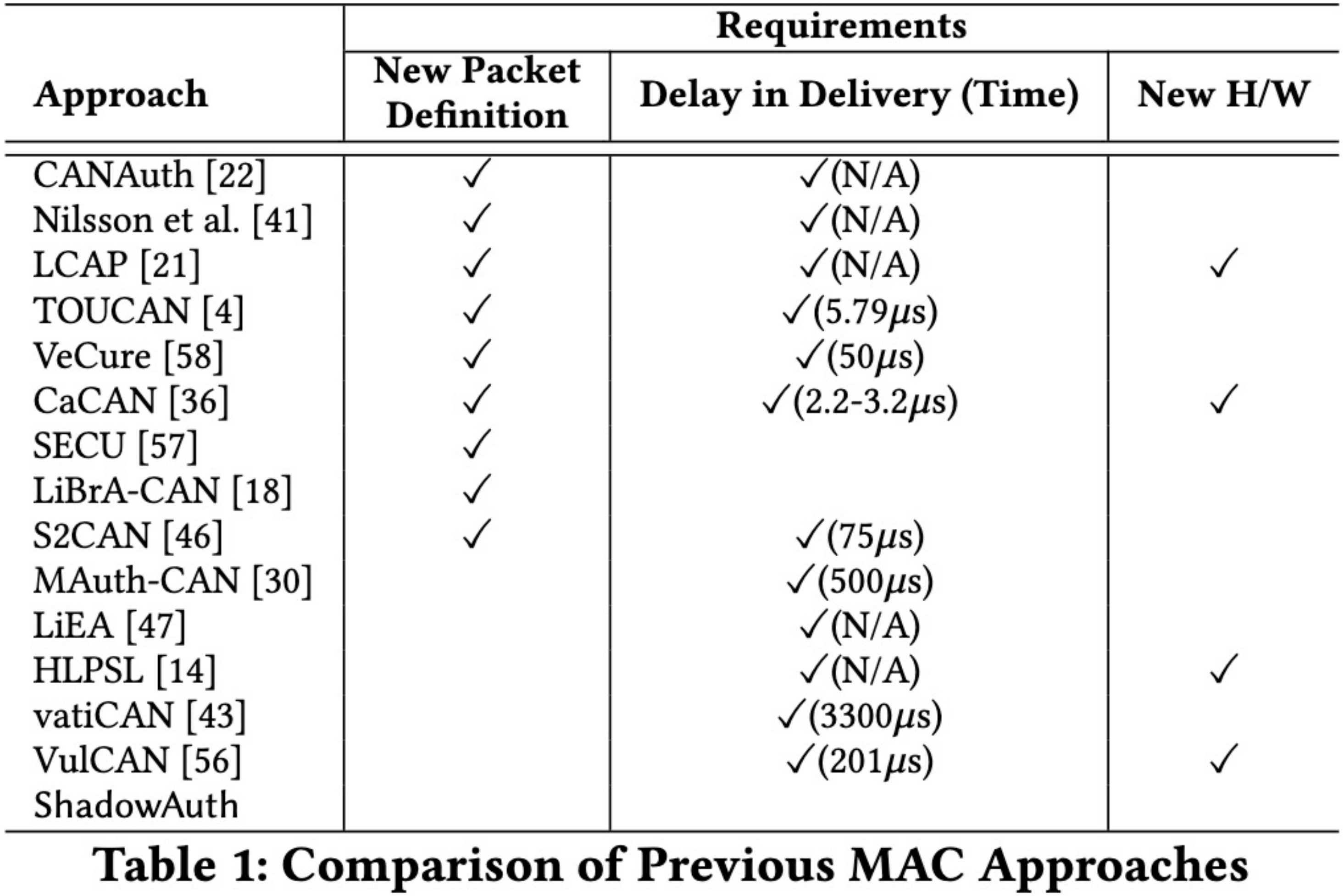
Evaluation: Asynchronous Delay





IDS	Trusted Base	False Positive (%)	False Negative (%)
CIDS [8]	Clock skew	0.055	0
VIDEN [9]	Voltage level	0.2	0.2
EASI [33]	Voltage edge	0	0.03

Table 3: Attack Detection Comparison with IDSs



Approach	Requirements		
	New Packet Definition	Delay in Delivery (Time)	New H/W
CANAuth [22]	✓	✓(N/A)	
Nilsson et al. [41]	✓	✓(N/A)	
LCAP [21]	✓	✓(N/A)	✓
TOUCAN [4]	✓	✓(5.79μs)	
VeCure [58]	✓	✓(50μs)	
CaCAN [36]	✓	✓(2.2-3.2μs)	✓
SECU [57]	✓		
LiBrA-CAN [18]	✓		
S2CAN [46]	✓	✓(75μs)	
MAuth-CAN [30]		✓(500μs)	
LiEA [47]		✓(N/A)	
HLPSL [14]		✓(N/A)	✓
vatiCAN [43]		✓(3300μs)	
VulCAN [56]		✓(201μs)	✓
ShadowAuth			

Table 1: Comparison of Previous MAC Approaches