The Feasibility of High-Performance Message Authentication in Automotive Ethernet Networks

VehicleSec 2023, San Diego

Evan Allen, Zeb Bowden, Randy Marchany, J. Scot Ransbottom

Virginia Tech

Issue: Lack of message authentication within in-vehicle networks (IVNs)

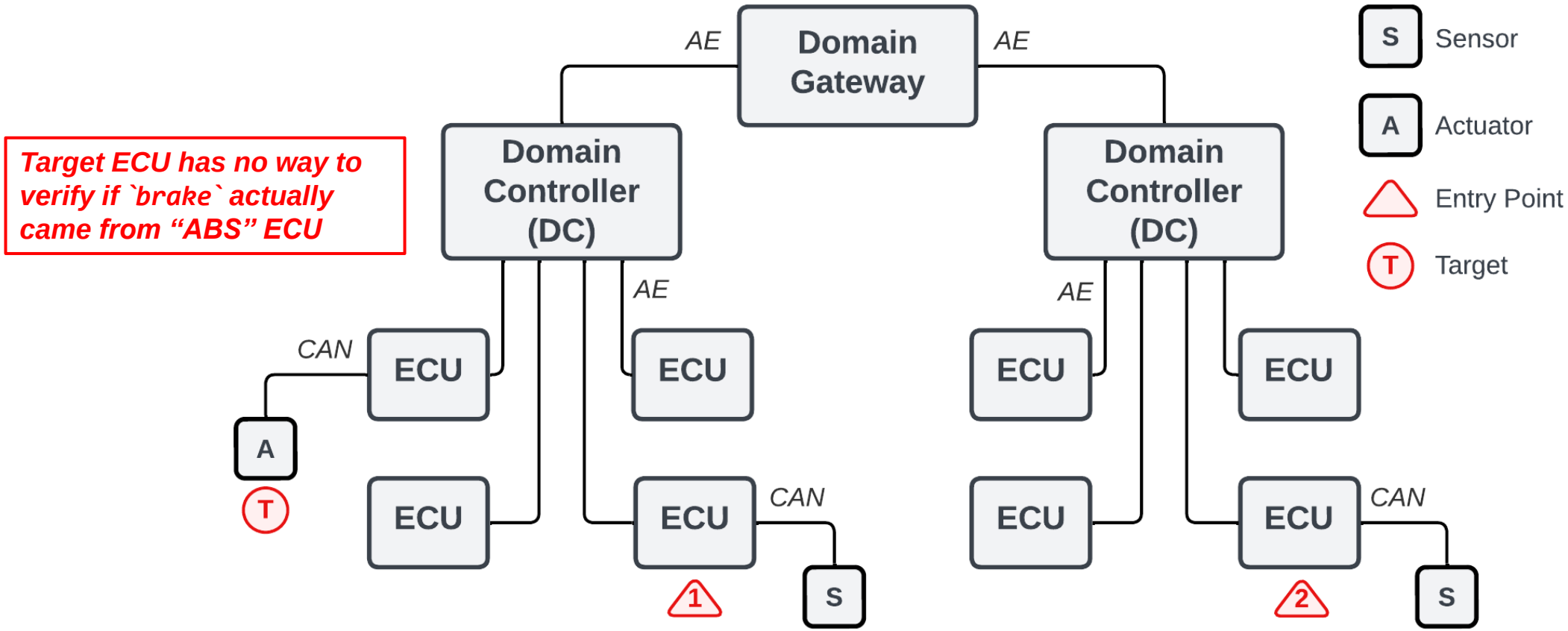
- IVNs such as CAN, FlexRay, Automotive Ethernet have no standard message authentication mechanisms

- Means malicious ECUs can spoof messages from other ECUs
  - “Hit the brakes!” ~ from an ECU that shouldn’t say that

- We need some way for an ECU to be able to verify the integrity and source of a message.

Threat Model

(Example Architecture)



How to authenticate messages?

- Message Authentication Codes (MACs)
  - Conventional solution from IT world
  - Cryptographic tag appended to message that verifies the integrity and source of the message

- Takes time to compute / verify!

TABLE I. PERFORMANCE REQUIREMENTS FOR VARIOUS CLASSES OF DATA, ADAPTED FROM [3], [13], [23]

Data Class	Throughput (Mbps)	Max. Latency (ms)	Period (ms)
Critical control	0.5-1	0.1	Event driven
Normal control	0.5-1	5-50	5-50
Radar	0.1-15	10	10
Ultrasonic	0.01-0.23	20	20
Camera Video*	~52	33	33
Lidar	20-100	10	10

*30 frames per second, compressed

Question:

Is it possible to do MAC authentication at these speeds?

Existing Work

- Two separate challenges:
  - very low latency & very high throughput
- Could not find any papers that could do <0.1ms latency, >80Mbps speeds in software, though
- Pena et al. achieved <0.1ms latency and >200Mbps speeds with MACsec using an FPGA (hardware)

Gatekeeper Latency Profiling

- Profiled one recent authentication proposal

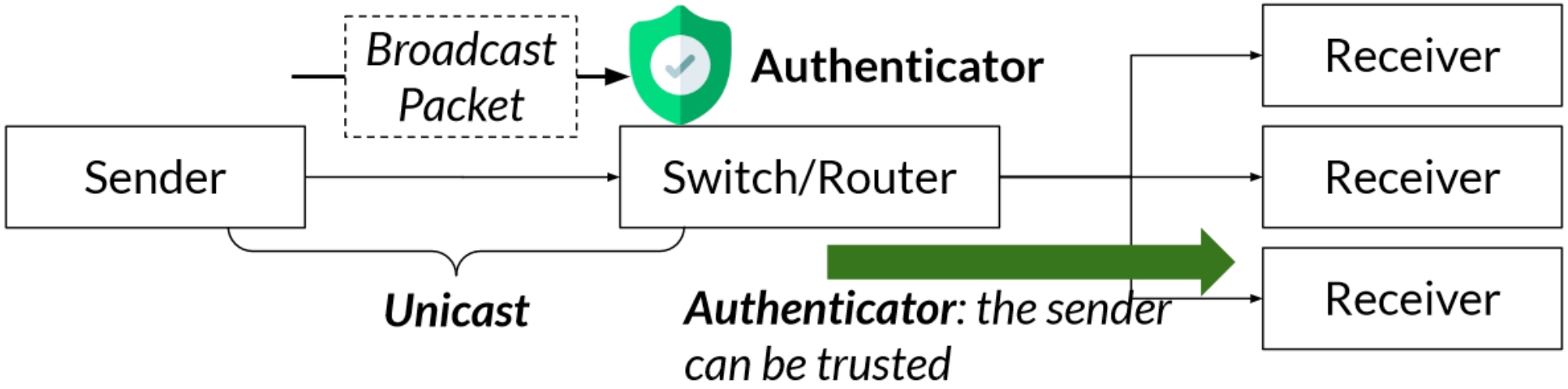
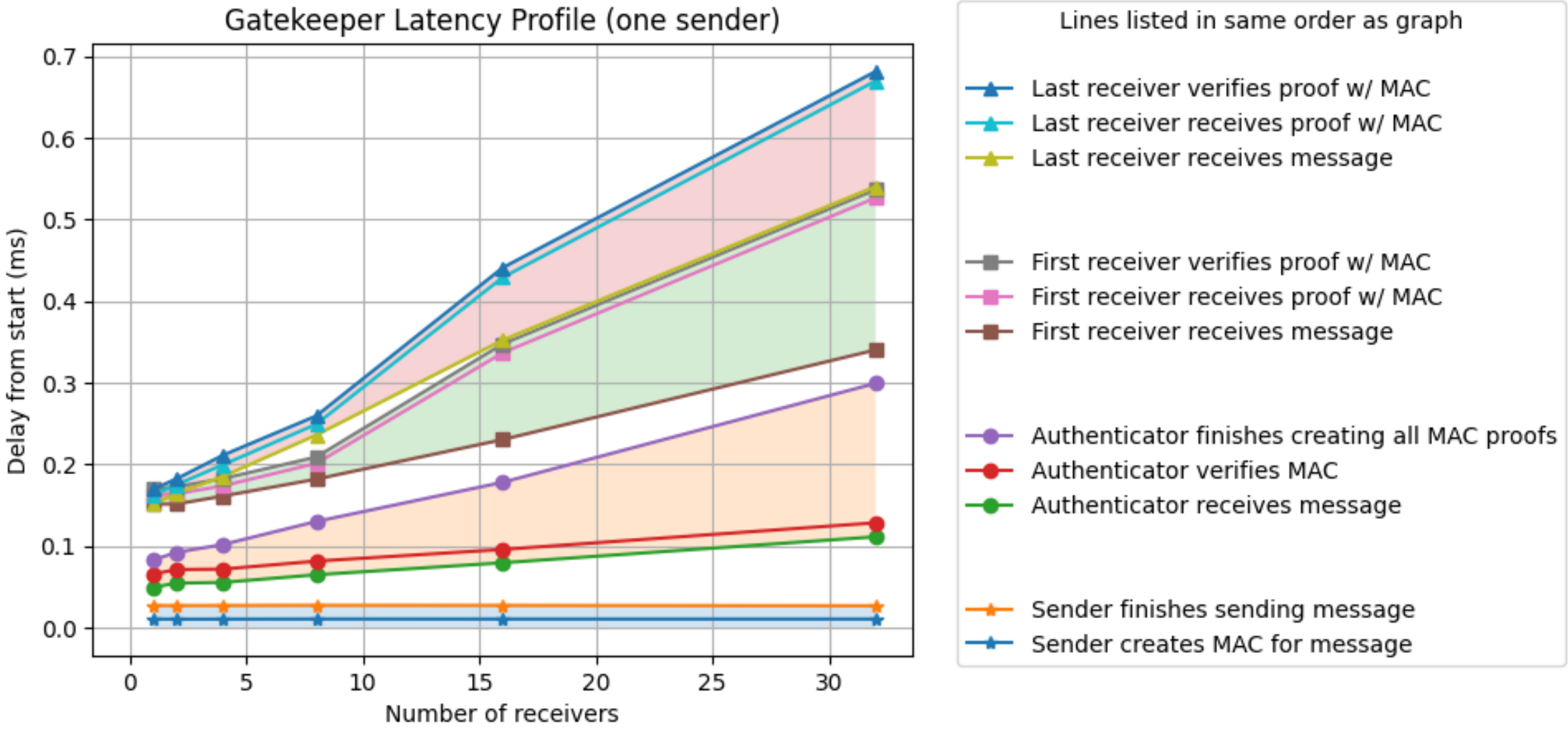


Figure 2: The overview design of GATEKEEPER.


Gatekeeper Latency Profile (one sender)
Delay from start (ms)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
25
30
Number of receivers
Lines listed in same order as graph
Last receiver verifies proof w/ MAC
Last receiver receives proof w/ MAC
Last receiver receives message
First receiver verifies proof w/ MAC
First receiver receives proof w/ MAC
First receiver receives message
Authenticator finishes creating all MAC proofs
Authenticator verifies MAC
Authenticator receives message
Sender finishes sending message
Sender creates MAC for message

Throughput Issues

- Gatekeeper authors found that their development board couldn't run hashing functions / encryption functions fast enough
- If those aren't fast enough, how could any protocol be fast enough?

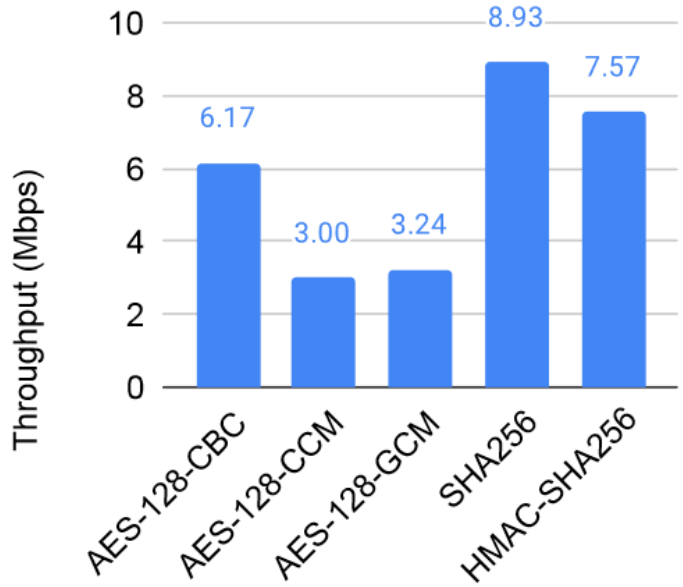


Figure 6: Performance of symmetric cipher suites and hash functions on the development board.

Discussion

Is it possible to do MAC authentication at these high speeds?

- Most likely only with dedicated hardware – software is not fast enough
- That gets expensive! Especially if every ECU needs it

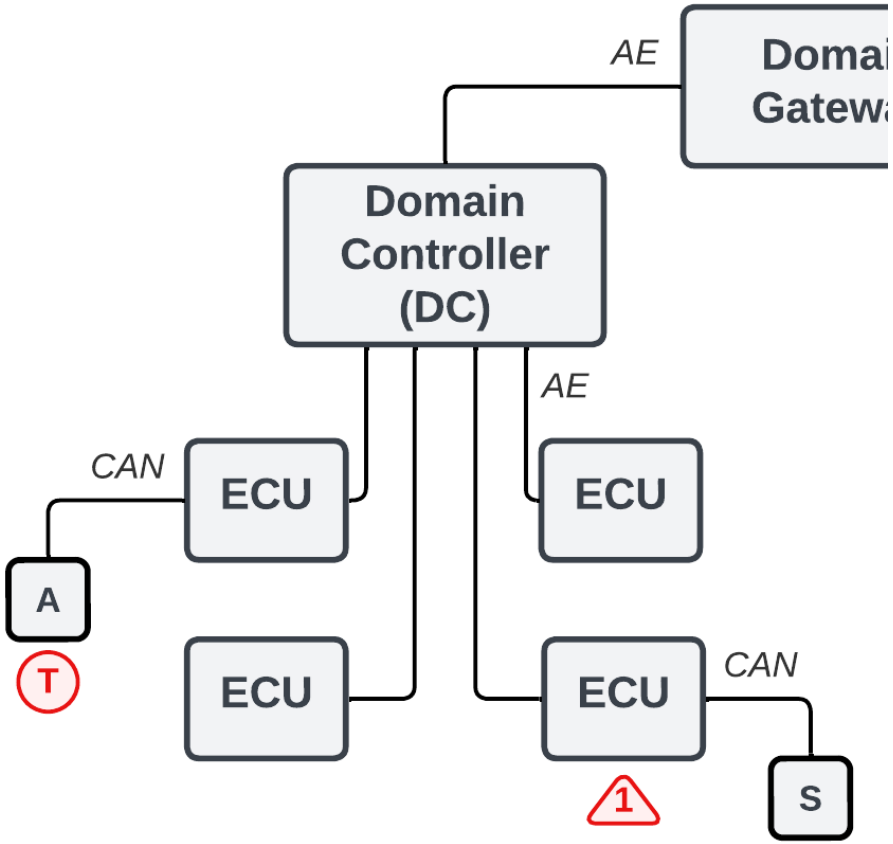
- Software methods like Gatekeeper still OK for lower-performance applications, like ultrasonic data (<0.23 Mbps, 20ms)

We still need a solution for low-latency, high-throughput data…

Okay, now what?

Idea: Reduce need for cryptographic authentication

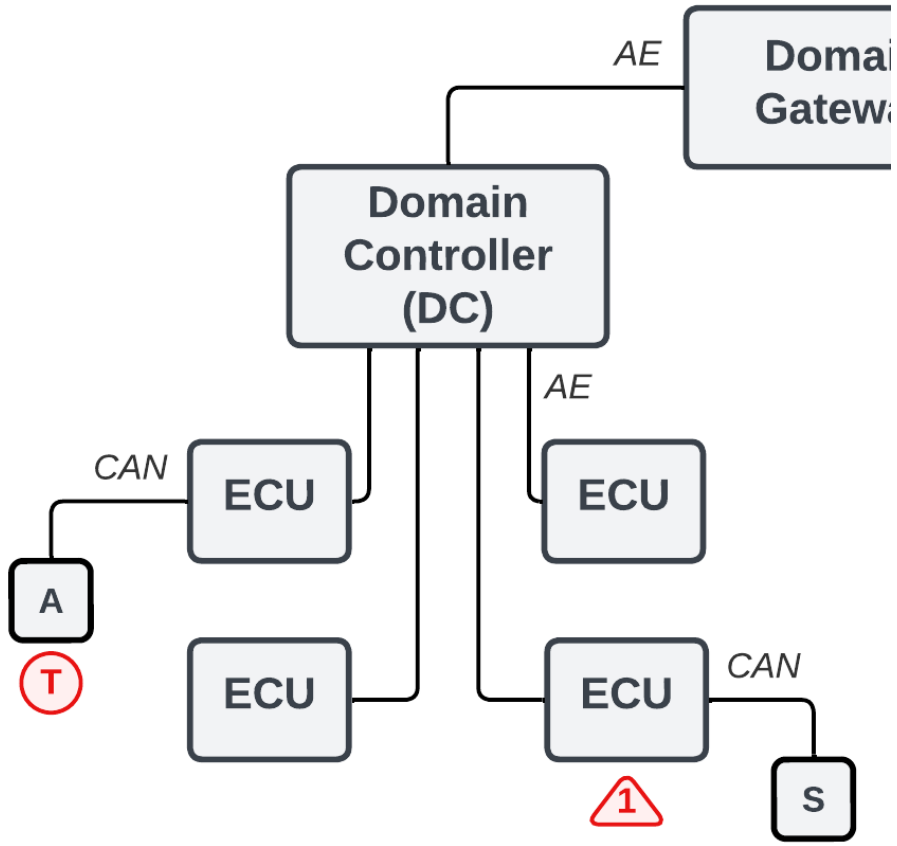
- How can we stop message spoofing without MACs or cryptography?
- Maybe we can use hardware ports.
  - Usually a bad idea in IT, but vehicle networks are different
- **Assumptions:**
  - **No physical man-in-the-middle**
  - **DC not compromised *****



Idea: ECUs implicitly trust traffic; domain controller does security work

Because ECUs don’t share a common bus…

- ECUs can trust that all incoming messages are from the domain controller

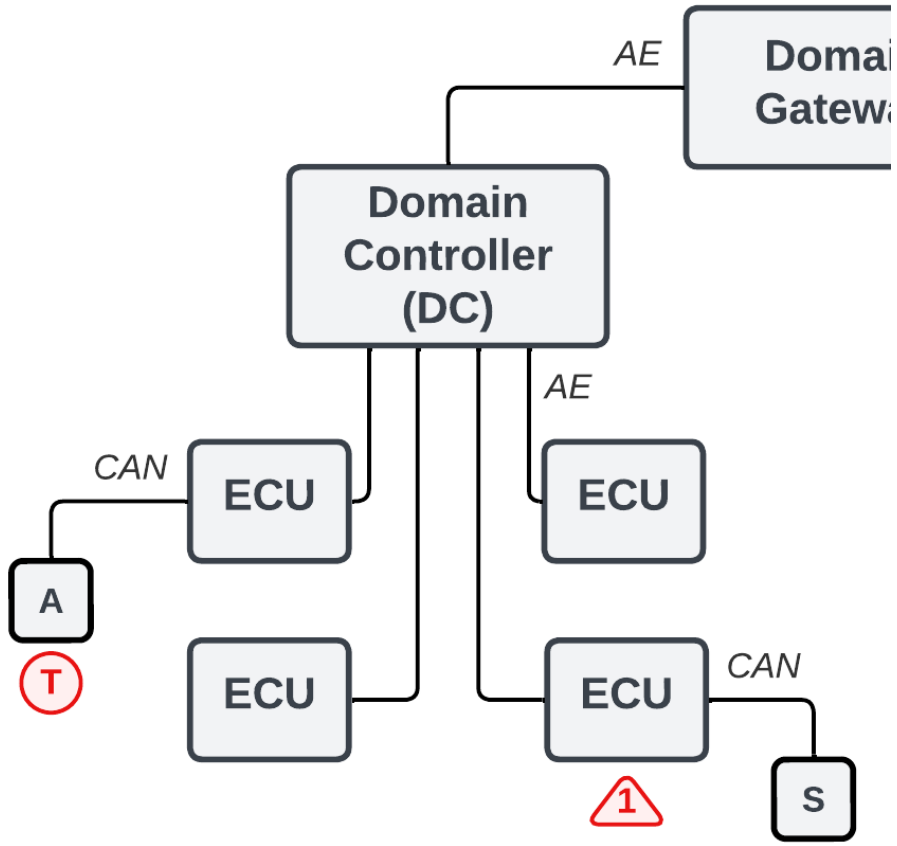


Idea: ECUs implicitly trust traffic; domain controller does security work

Because ECUs don't share a common bus…

- ECUs can trust that all incoming messages are from the domain controller

- Domain controller can trust that messages on a hardware interface are from that ECU
  - (even if the ECU is compromised)

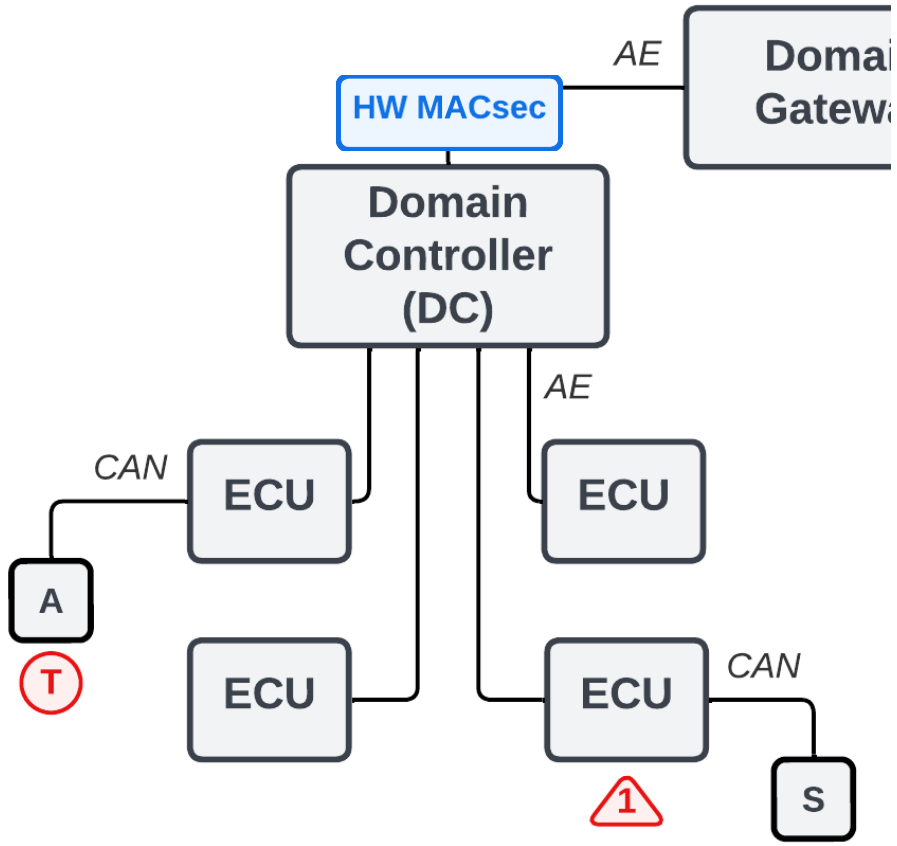


Idea: ECUs implicitly trust traffic; domain controller does security work

- Intra-domain traffic is thus authenticated.

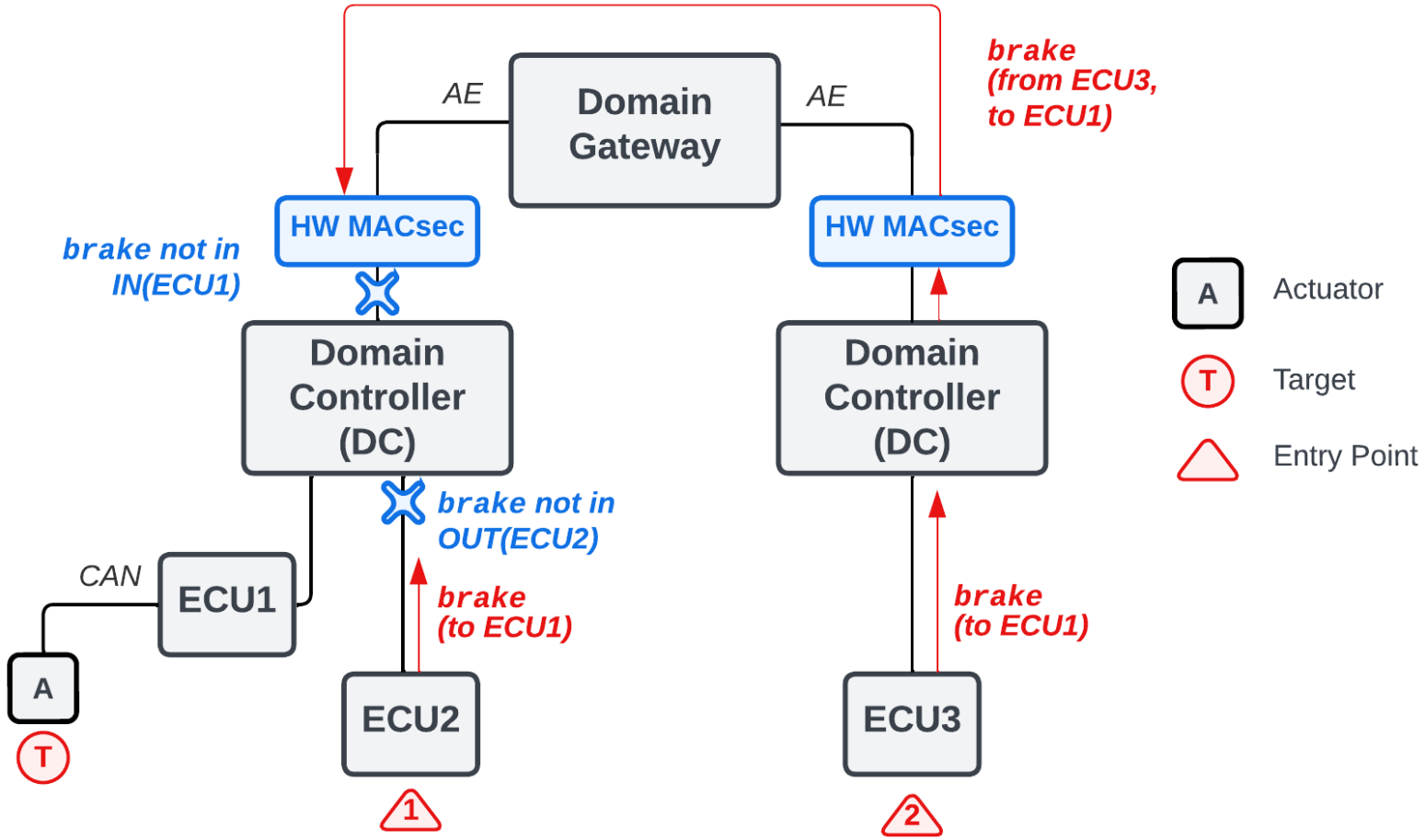
- What about *inter-domain* traffic?
- Here we can use hardware MACsec
  - Cost-feasible to implement for just a few domain controllers
  - Demonstrated to be fast enough earlier

- Result: DC knows actual sender / receiver of all traffic



Idea: DC can act as firewall given security policy

- Create **security policy**.
- Define what message types each ECU E may:
  - Send $(OUT(E))$
  - Receive $(IN(E))$
- DC blocks traffic violating these rules
- ECUs do no work!



Limitations / Areas for Improvement

- Assumes domain controller (DC) is not compromised (***)
  - Tradeoff for speed and cost
  - Could spend more resources on securing DC
  - Common assumption in other work (e.g., Gatekeeper)
  - *How could we mitigate this risk?*

- Assumes no physical man-in-the-middle
  - If an attacker had physical access to the vehicle, could just cut brake lines

Future Work

- Flesh out domain controller firewall approach, build prototype
- Investigate performance, limitations
  - Can it satisfy the previous performance requirements?
  - How restrictive can the `in` / `out` policies be?
  - How much overhead do they cause?

- Reproduce MACsec benchmark results, determine if it can stay performance compliant on low-cost hardware.
  - Could make MACsec more accessible to manufacturers.

Questions?