Security requirements for vehicle security gateways

Ben Gardiner (NMFTA), John Maag (Cummins), Dr. Ken Tindell (JK Energy)

CAN security gateways are commonly deployed to keep untrusted parts of a vehicle away from the trusted part. Although conceptually simple, they often introduce subtle problems that can escape detection in testing and only manifest after a vehicle is deployed or experiences a sophisticated security attack. In particular, there are common problems with the handling of transmitted frames that can lead to the intermittent failure of cryptographic protections, loss or corruption of messages and disruption of the traffic on the protected CAN bus. Avoiding these problems are part of the requirements for a robust security gateway.

1. Introduction

A security gateway is a common technique for securing a CAN bus (alongside cryptographic messaging, intrusion detection and hardware security [1]). The main purpose is to protect the hard real-time mechatronic control buses from being disrupted by vulnerable devices with wireless connections, such as telematic control units (TCUs) and in-vehicle infotainment (IVI) systems. This is necessary because when (and for sufficiently complex systems it is 'when' not 'if' [2]) those devices are compromised in an attack, the attacker cannot compromise the fundamental controls of a vehicle.

There is a second use of a CAN security gateway in a vehicle network: to allow a CAN bus to be partitioned into segments so that if one of the segments becomes physically compromised then other segments can continue to operate without disruption. For example, the CAN Injection attack [3] gains access to a CAN bus via an easily reached part of the car (e.g. a headlight connector) allowing a theft device to spoof messages (e.g. commands to disable the immobilizer or unlock the doors). A security gateway between an easily accessed CAN bus and the rest of the car would prevent this.

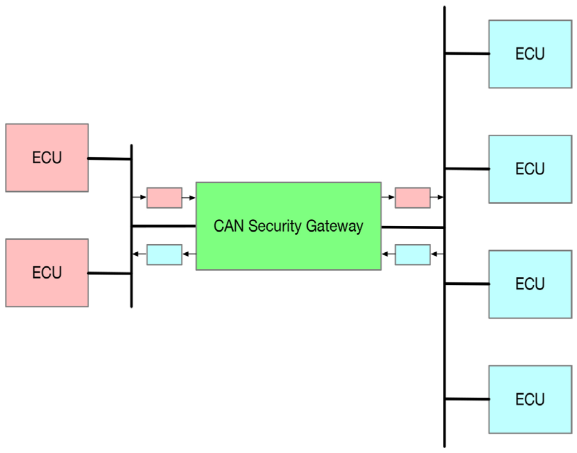

Figure 1: Conceptually simple CAN security gateway

A common way to define security is through the 'CIA Triad': Confidentiality, Integrity, Availability. For CAN bus, the most common attack is an integrity attack: the CAN ID is chosen by an attacker to send forged messages so that receivers will act upon it as if it were genuine.

And so a CAN security gateway is conceptually simple: a device that connects to two CAN buses and filters frames so that only the appropriate subset are copied from one bus to the other (Figure 1).

But there are many subtle details, especially with respect to availability, that if not handled correctly result in an insecure or nonfunctioning network. The National Motor Freight Transport Association (NMFTA) in the US developed a set of requirements for security gateways to ensure correct behavior. These requirements are defined in a formal requirements language and have been made public [5]. This paper draws on those requirements and focuses on two vital aspects of a CAN security gateway: 1) Frame transmission and 2) Frame dropping.

2. Frame Transmission

The transmission of CAN frames might appear simple but there are subtle pitfalls. The first of these surrounds the order of transmission. It is very important that a CAN security gateway does not re-order frames because the order of events can be very important for applications. For example, the ISO-TP [6] transport protocol assembles a large message from a sequence of CAN frames, and if frames are re-ordered then the reception of the whole message can fail (there are sequence numbers attached to each CAN frame, but some implementations simply check for a gap and then trigger an error, and some even ignore the sequence number).

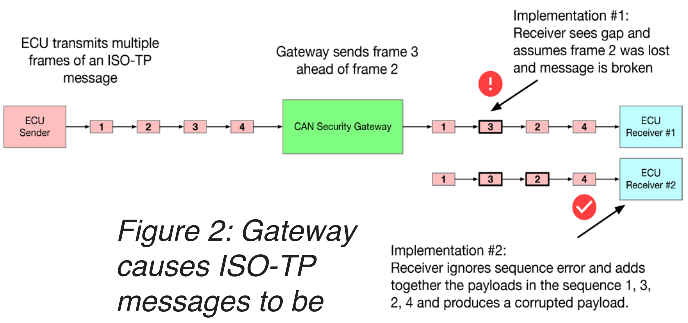
ECU transmits multiple frames of an ISO-TP message
Gateway sends frame 3 ahead of frame 2
Implementation #1: Receiver sees gap and assumes frame 2 was lost and message is broken
ECU Sender
1
2
3
4
CAN Security Gateway
1
3
2
4
ECU Receiver #1
1
3
2
4
ECU Receiver #2
Implementation #2: Receiver ignores sequence error and adds together the payloads in the sequence 1, 3, 2, 4 and produces a corrupted payload.

Figure 2: Gateway causes ISO-TP messages to be lost or corrupted

This problem also occurs when frames are cryptographically protected, for example with SecOC [7] or CryptoCAN [8]. Cryptographic schemes are required to protect against a Replay Attack [9], where an attacker copies a legitimate frame and re-sends it later to get a receiver to act upon it. These schemes will typically see a re-ordered frame as an attempted replay attack, and this will often result in a false alarm, and too many of these will result in real attacks being ignored.

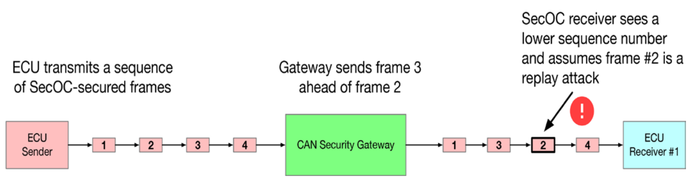
ECU transmits a sequence of SecOC-secured frames
Gateway sends frame 3 ahead of frame 2
SecOC receiver sees a lower sequence number and assumes frame #2 is a replay attack
ECU Sender
1
2
3
4
CAN Security Gateway
1
3
2
4
ECU Receiver #1

Figure 3: A gateway can cause a replay attack false alarm

CAN frames are typically re-ordered not by security gateway software explicitly written to do this but by CAN controller hardware. Most CAN controllers have a transmit priority queue, where the next frame in the queue is not the head of the queue but the frame with the lowest CAN ID. Some controller hardware implements this with a set of comparator circuits comparing the ID of each frame buffer slot, and some hardware implements this with a state machine that sequentially scans down the frame buffer slots, keeping track of the lowest numeric value seen so far. In any case, there is a problem when there are two or more CAN frames in the queue with the same ID: the hardware must decide which of these to send first. In most cases, this ‘tie’ is resolved arbitrarily (such as the number of the frame buffer slot, or the last frame seen in the sequential scan).

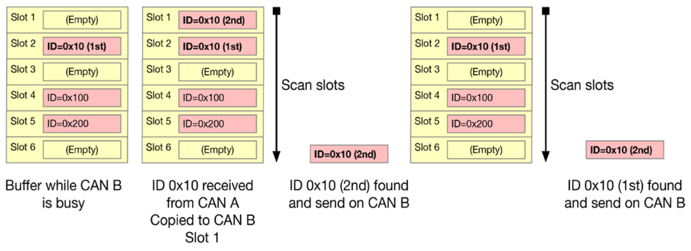

Figure 4: Sequence of transmit queue states, frames with ID 0x10 are transmitted out of order

One reason for two frames with the same ID being in the transmit queue at the same time is because of Frame Arrival Jitter. In general, jitter is the variability of the specific timing of a
periodic event. Most CAN frames in a vehicle control network are generated periodically (often they carry data produced by a periodic control loop in an ECU). But although they are generated strictly periodically (i.e. there is a fixed time between generating one frame and the next one), they do not

arrive at receivers with the same timing: the underlying period is the same but the arrival time has jitter (Figure 5).

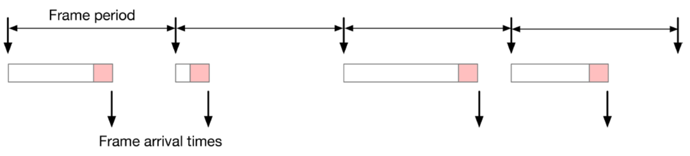

Figure 5: Frames are queued periodically but arrival times are jittered due to variable times waiting to win arbitration

Jitter means that if periodic frames are simply copied from an incoming CAN controller FIFO
to an outgoing priority queue then more than one frame with the same ID can be in the queue at the same time (Figure 6).

Frame jitter is a particularly difficult problem because the jitter of a given frame observed at run time depends on the specific traffic pattern on the two CAN buses. During testing there may be no observed re-ordering of CAN frames but under particular loading of the two buses it may become apparent by highly intermittent failures of security or ISO-TP messaging. Tracing the cause of such faults may not even be possible in a post-production environment, and in any case the fault is not due to a simple bug but a design flaw in the security gateway.

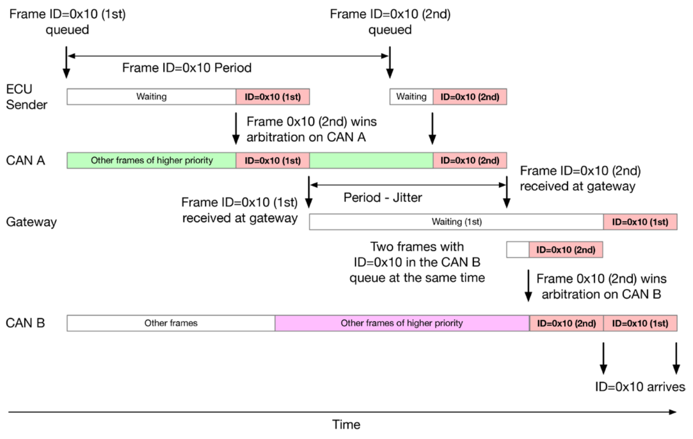
Frame ID=0x10 (1st) queued
Frame ID=0x10 (2nd) queued
Frame ID=0x10 Period
ECU Sender
Waiting
ID=0x10 (1st)
Waiting
ID=0x10 (2nd)
Frame 0x10 (2nd) wins arbitration on CAN A
CAN A
Other frames of higher priority
ID=0x10 (1st)
ID=0x10 (2nd)
Frame ID=0x10 (2nd) received at gateway
Period - Jitter
Frame ID=0x10 (1st) received at gateway
Gateway
Waiting (1st)
ID=0x10 (1st)
Two frames with ID=0x10 in the CAN B queue at the same time
ID=0x10 (2nd)
Frame 0x10 (2nd) wins arbitration on CAN B
CAN B
Other frames
Other frames of higher priority
ID=0x10 (2nd)
ID=0x10 (1st)
ID=0x10 arrives
Time

Figure 6: Frame jitter causing frame re-ordering

At first sight, the solution to the re-ordering problem is to adopt FIFO buffering in the transmit queue. However, this leads directly to another problem: CAN Priority Inversion [4]. Priority inversion occurs when the front of a FIFO queue contains a low priority frame, with urgent high priority frames behind it. The frame at the front of the queue will not win arbitration for a long time if the bus becomes busy for a long time with higher priority traffic, and thus the urgent frame is delayed for a long time (Figure 7).

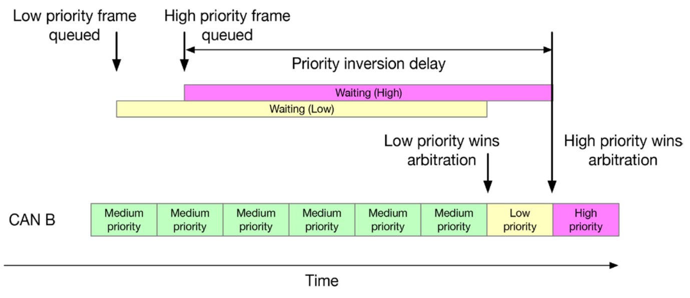

Figure 7: Priority inversion on CAN bus

As for frame jitter, priority inversion can be very intermittent: it might not show up in testing because it depends on the particular sequence of CAN frames being queued and transmitted on the bus. But the consequences are no less severe: an urgent 10ms period frame might sometimes be delayed for 90ms, and this could lead to all kinds of failures (including triggering timeouts where a component was assumed to have failed). The consequences of such failures in a production vehicle could be severe.

The resolution of this seeming contradiction between the need for both priority and FIFO transmission is to observe that the requirement for FIFO transmission is with respect to frames with the same ID. This means that the correct frame transmission policy in a gateway is not to have just a priority queue but one that is fed by a FIFO for each frame ID (Figure 8).

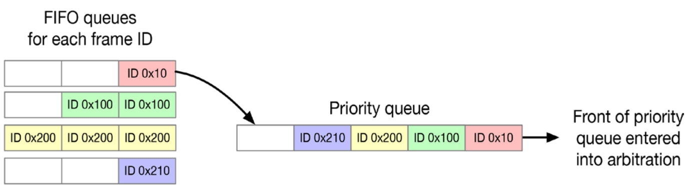

Figure 8: FIFO queues feeding into a priority queue

There is one further problem that frame jitter causes with respect to a security gateway: it can damage the real-time properties of the destination bus. To see why, look again at Figure 6. The arrival jitter from CAN A causes frames to 'bunch up' in time, and by queueing them on the CAN B immediately, the security gateway has caused two frames to be queued in a much shorter time than they were on CAN A. This means that over a short time interval, the bus load due to the forwarded frame is much higher (over a long interval the bus load is of course the same, but real-time systems are ones where short-term utilisation is important).

By deferring a frame until the expiry of its period time, the timing behavior on the destination bus is no different than if that frame were queued directly by the original transmitter ECU. The only difference is that deferral adds to the latency of the second frame in this scenario. But this additional latency through the gateway does not cause the *worst-case* latency to rise: in effect, jitter causes frames to appear earlier than they might have and the deferral mechanism only applies to early frames. By deferring early frames, the security gateway can maintain

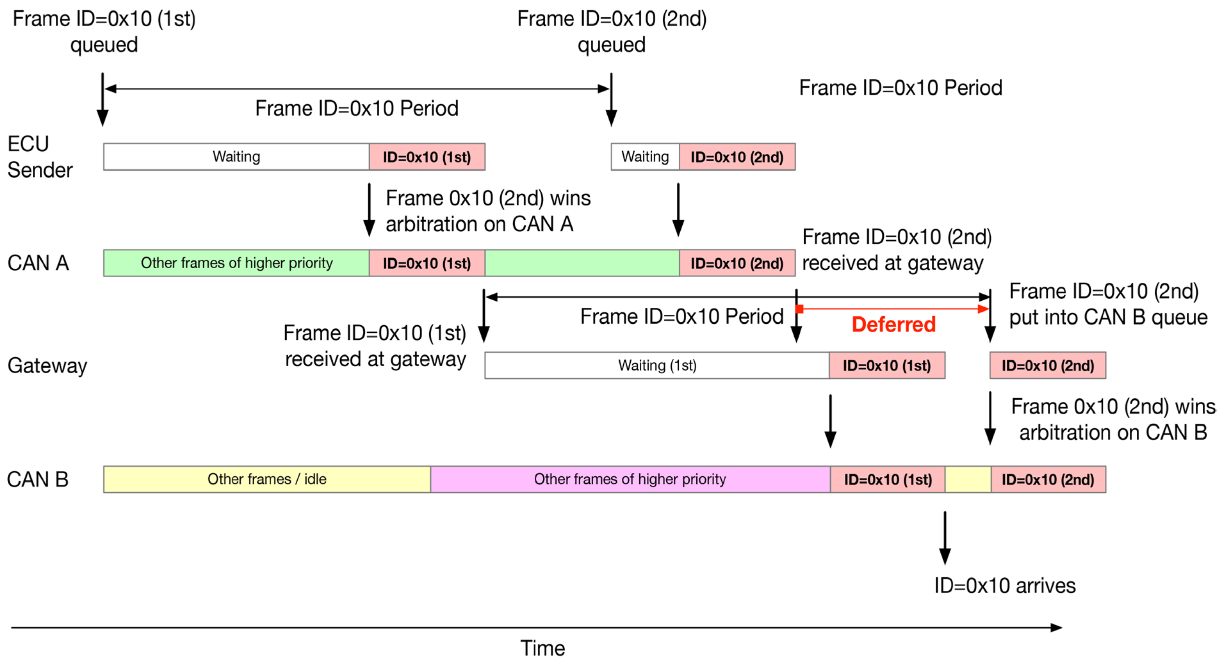
Frame ID=0x10 (1st) queued
Frame ID=0x10 (2nd) queued
Frame ID=0x10 Period
Frame ID=0x10 Period
ECU Sender
Waiting
ID=0x10 (1st)
Waiting
ID=0x10 (2nd)
Frame 0x10 (2nd) wins arbitration on CAN A
Frame ID=0x10 (2nd) received at gateway
CAN A
Other frames of higher priority
ID=0x10 (1st)
ID=0x10 (2nd)
Frame ID=0x10 Period
Deferred
Frame ID=0x10 (2nd) put into CAN B queue
Frame ID=0x10 (1st) received at gateway
Gateway
Waiting (1st)
ID=0x10 (1st)
ID=0x10 (2nd)
Frame 0x10 (2nd) wins arbitration on CAN B
CAN B
Other frames / idle
Other frames of higher priority
ID=0x10 (1st)
ID=0x10 (2nd)
ID=0x10 arrives
Time

Figure 9: Deferring early frame arrivals

This higher short-term bus load caused by a security gateway means that an attacker could craft a traffic pattern on a CAN bus and induce a timing fault on a CAN bus protected by a security gateway, leading to consequent failures (such as missed timeouts, as discussed above). In the CIA Triad this is an example of a failure of availability – a type of denial-of-service attack.

To defend against this type of attack (and to eliminate intermittent timing faults due to jitter) the security gateway should implement frame deferral: a frame should not be put into its transmit FIFO until at least one frame period time has elapsed since the previous frame was queued (Figure 9).

worstcase latency guarantees made by existing CAN timing analysis [10] and so maintain end-to-end latency guarantees from one CAN bus to another.

3. Frame dropping

CAN is unique amongst the common field buses by providing Atomic Broadcast. This is an incredibly useful property for building robust systems, and many applications rely on it (even if unwittingly). When a message is marked as 'sent' at a transmitter then the transmitter knows that every online node received a valid copy of that message. In effect, CAN implements consensus in hardware. With other protocols, such as Ethernet, a message that is received incorrectly (e.g. the CRC – or ‘FCS’ for Ethernet frames – does not match) then the frame is discarded. This means that simple noise on the bus can lead to divergence between nodes in the view of the state of a system (one receiver sees only an old sensor reading and another a newer one). Distributed consensus is a key building block of robust distributed real-time control systems, and obtaining this property with software protocols on top of a non-robust fieldbus is difficult. Indeed, Lamport’s Byzantine Generals Problem arose from a research project trying to do that [11].

This leads to an important requirement for a security gateway: to maintain CAN’s distributed consensus property. Introducing a security gateway should not result in breaking the assumptions of existing applications, especially if those assumptions are implicit. In other words, a security gateway must not drop frames (unless of course there is a genuine fault in the system, where ‘fault’ includes an attack on the system).

A security gateway is, of course, obliged to drop frames in order to defend against an attack. For example, diagnostic tester frames should be dropped unless there is a legitimate diagnostic tester attached, frames in a flood attack should be dropped, etc. But a security gateway must also not drop frames due to insufficient buffer space in a transient overload.

On the receive side, this means that frames must be received and processed at the full rate of the CAN bus: if two frames with the same ID arrive back-to-back then the CAN controller hardware and its driver software must not lose one of those frames. Generally this means handling CAN frames via interrupt service routines (ISRs), and the scheduling of the CPU must be able to handle all interrupts from all CAN controllers within their respective deadlines (as well as all other demands on the CPU). This will typically require careful scheduling of the CPU and knowledge of the worst-case execution times of ISRs. For the transmission of CAN frames, there must be enough space to store all pending frames (including deferred frames). One technique for bounding the buffer space is, like bounding stack space in ECU software, to keep increasing it until it appears there are no more overflows. Obviously this is a poor technique because it relies upon having seen the worst-case scenario during testing (and therefore very unlikely to have observed the worst-case pattern of traffic on both CAN buses simultaneously). It is better to instead bound the buffer space by calculating the longest time a given frame can take in its queue (i.e. its worst-case latency on the CAN destination bus), the time it can deferred but stored, and the maximum number of times it could legitimately be received from the source CAN bus during these times.

4. Summary and conclusions

A security gateway has to meet non-functional requirements to adequately defend a CAN bus from attack and also not to introduce faults into the system. The NMFTA has enumerated all the requirements for such a gateway, in a formal description language, and has made these publicly available.

There are very specific requirements of a security gateway to protect the real-time and distributed consensus properties of CAN, and these demand the careful handling of the ordering and buffering of CAN frames for transmission to avoid pitfalls of common CAN controller hardware designs and frame arrival jitter on CAN buses. Not doing this at the design stage may lead to faults during transient overload conditions that are not feasible to observe during testing but will likely manifest during the millions of collective hours of run-time in production vehicles, leading to problems with reliability and security that could cause business level failures. The ability to bound the real-time behavior of the CAN bus is a vital component in ensuring that problems can be found analytically during development and not left for others to discover in the field.

References

[1] Defending The CAN Bus: Security Gateways (https://kentindell.github.io/2021/11/24/can security-part-3)
[2] Pwn2Own Automotive 2024 (https://vicone.com/pwn2own-automotive)
[3] CVE-2023-29389 (https://nvd.nist.gov/vuln/detail/CVE-2023-29389)
[4] CAN Priority Inversion (https://kentindell.github.io/2020/06/29/can-priority-inversion)
[5] Implementation Requirements for Secured Gateways, NMFTA (https://nmfta.org/whitepaper/implementation-requirements-for-secured-gateways)
[6] ISO 15765-2:2016 Road vehicles, Diagnostic communication over Controller Area Network (DoCAN) Part 2: Transport protocol and network layer services
[7] Specification of Secure Onboard ommunication Protocol, AUTOSAR 969 R23-11 2023-11-23
[8] Securing CAN: Introduction to CryptoCAN, CAN Newsletter December 2022, (CAN in Automation)
[9] Replay attack (https://en.wikipedia.org/wiki Replay_attack)
[10] Guaranteeing Message Latencies on Controller Area Network (CAN), K. Tindell and A. Burns, Proceedings 1st International CAN Conference, 1994
[11] The Byzantine Generals Problem, L. Lamport, R. Shostak, M. Pease, ACM Transactions on Programming Languages and SystemsVolume 4 Issue 3, 1982

Ben Gardiner, NMFTA
1001 N. Fairfax St., Ste. 600
US-22314 Alexandria, VA
www.nmfta.org

John Maag, Cummins
1900 McKinley Ave. MC 50197
US-47201 Columbus IN
www.cummins.com

Dr. Ken Tindell, JK Energy
Dutch House, Mustow Street
GB-IP33 1XL Bury St Edmunds, Suffolk
www.jkenergy.com