Extending AUTOSAR's Counter-based Solution for Freshness of Authenticated Messages in Vehicles

Extending AUTOSAR's Counter-based Solution for Freshness of Authenticated Messages in Vehicles

Thomas Rosenstatter
Chalmers University of Technology
Gothenburg, Sweden
Email: thomas.rosenstatter@chalmers.se

Christian Sandberg
Volvo Trucks
Gothenburg, Sweden

Tomas Olovsson
Chalmers University of Technology
Gothenburg, Sweden
Email: tomas.olovsson@chalmers.se

***Abstract*—Nowadays vehicles have an internal network consisting of more than 100 microcontrollers, so-called Electronic Control Units (ECUs), which control core functionalities, active safety, diagnostics, comfort and infotainment. The Controller Area Network (CAN) bus is one of the most widespread bus technologies in use, and thus is a primary target for attackers. AUTOSAR, an open system platform for vehicles, introduced in version 4.3 SecOC Profile 3, a counter-based solution to provide freshness in authenticated messages to protect the system against replay attacks. In this paper, we analyse and assess this method regarding safety constraints and usability, and discuss design considerations when implementing such a system. Furthermore, we propose a novel security profile addressing the identified deficiencies which allows faster resynchronisation when only truncated counter values are transmitted. Finally, we evaluate our solution in an experimental setup in regard to communication overhead and time to synchronise the freshness counter.**

***Index Terms*—cyber-physical systems, security, message authentication, freshness, automotive.**

1. Introduction

The internal network of modern vehicles consists of more than 100 Electronic Control Units (ECUs) that communicate via different network technologies, such as Controller Area Network (CAN) and Ethernet. CAN is a relatively old technology developed by Robert Bosch GmbH in 1983 and later standardised in ISO 11898 [4]. Back then, securing the CAN bus was not an issue, as listening and transmitting on the bus required special equipment and physical access to the vehicle. This assumption changed when vehicles became connected to the outside world, e. g., the internet, mobile phones, and other vehicles.

Miller and Valasek demonstrated a remote exploitation back in 2015 [6], where they were able to remotely control a vehicle over the Internet. Given this, and other successful attacks in more recent years underlines the need for security in the automotive domain, especially, a secure transmission of data.

AUTOSAR, a system architecture developed by a consortium of vehicle OEMs and suppliers, defines in *Specification of Secure Onboard Communication v4.4.0* [1] three so-called SecOC Profiles, which provide message authentication to ensure that messages were sent by said origin and have not been altered during transmission. Messages are authenticated by calculating the corresponding Message Authentication Code (MAC) and sending it along with the clear text. SecOC Profile 2 verifies the authenticity of messages without an additional counter or timestamp, whereas SecOC Profile 1 and 3 use a counter-based Freshness Value (FV) in order to prevent replay attacks. The difference between the latter two is that SecOC Profile 3 includes a mechanism to synchronise the FV across senders and receivers. Such a mechanism to synchronise the FV is required when only a truncated FV is sent along with the data, for instance, due to the limited payload size and network technology used in automotive networks.

In this paper, we analyse and assess AUTOSAR's SecOC Profile 3 in detail, discuss design aspects and limitations of counter-based freshness solutions, and lastly propose a new profile that is independent of the underlying network architecture and addresses the identified issues. Additionally, we evaluate our method in terms of communication overhead and time to synchronise the FV.

2. AUTOSAR SecOC Profile 3 (JASPAR)

The AUTOSAR *Secure Onboard Communication Specification* [1] introduces SecOC Profile 3, also named JASPAR, a method that guarantees freshness and provides message authentication. Profile 3 applies CMAC/AES-128 [5] for message authentication and is used in combination with a master/slave synchronisation of a freshness counter as explained in [1, Annex A]. Profile 3 also introduces an entity called Freshness Value Manager (FVM), which is responsible for maintaining the current FV and synchronising the value between senders and receivers. The FVM can either run centrally on a dedicated ECU or decentralised in each sending ECU. A method for a fast synchronisation of the FV between the different entities is necessary, as the senders and receivers normally only store a portion of the FV in Non-Volatile Memory (NVM). In addition, a watchdog timer reset or other unexpected situations may occur where an ECU restarts and consequently fails to successfully verify

incoming Interaction Layer Protocol Data Units (I-PDUs) due to the fact that the FV has already changed.

In the following sections we describe the FV, the structure of the I-PDUs and continue with giving details under what circumstances Profile 3 is able to correctly verify the authenticity of I-PDUs with truncated FVs even when I-PDUs are lost.

2.1. Freshness Value and I-PDU Format

Profile 3 describes two configurations for sending data:

(1) sending the authenticated data, named *Authentic I-PDU*, and the authentication information, namely the *Cryptographic I-PDU*, separately; or
(2) sending the application data and the authentication information in one I-PDU, a *Secured I-PDU*, which is only possible if the data to be authenticated is short.

A truncated FV and MAC, which is further called authenticator, may be sent for each I-PDU since sending these values in full length significantly increases the network load. CAN, for instance, is already highly utilised and requires a truncation as the maximum payload size is 64 bits.

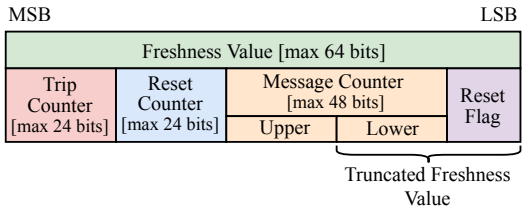
Freshness Value. The FV has a maximum size of 64 bits and consists of three sub-counters (their maximum size is given in braces): trip counter (24 bits), reset counter (24 bits) and message counter (48 bits), as shown in Figure 1a. Note that the length of the sub-counters must be adjusted individually and may not exceed the maximum size of the FV (64 bits). The trip counter is increased in units of trips, the reset counter is incremented periodically defined by a static parameter *ResetCycle* and the message counter is increased per message/I-PDU being sent. The reset flag shown in Figure 1a contains the n least significant bits of the reset counter.

Authentic I-PDU. This I-PDU shown in Figure 1b contains the data that is authenticated by the sender. Depending on the configuration it is sent apart from the authenticator and FV, or as part of the Secured I-PDU.

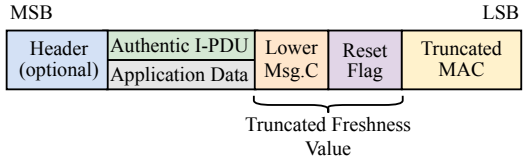
Secured I-PDU. Is generated when the Authentic I-PDU, the authenticator and the FV are combined in one I-PDU. It contains an optional header, the data to be transmitted, the truncated FV and the authenticator, as shown in Figure 1b. AUTOSAR recommends to transmit the lower 4 bits of the FV and the upper 28 bits of the MAC. Note that the authenticator contains the MAC of the plain text (the message) and the full length FV even in cases when only a truncated FV is transmitted.

Cryptographic I-PDU. Figure 1c gives details about this I-PDU which is sent along with the Authentic I-PDU and thus allows, in case of CAN, more bits of the FV and MAC to be transmitted.

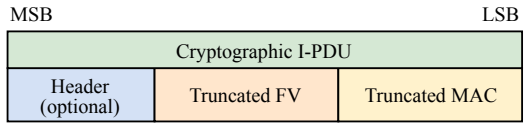
Synchronisation Message. Also named TripResetSyncMsg, is sent periodically by the FVM when either the trip counter increases or a new *ResetCycle* starts, for instance



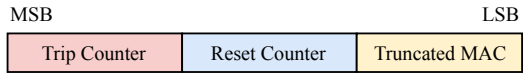
(a) Structure of the locally stored Freshness Value.



(b) Structure of the Secured I-PDU, i. e., containing both, data and authentication information.



(c) Structure of the Cryptographic I-PDU, i. e., containing only the authentication information.



(d) Structure of the Synchronisation Message.

Figure 1. Structure of the FV and I-PDUs [1, p.138,145].

every second. The synchronisation message, see Figure 1d, contains the trip and reset counter in their full length as well as the corresponding authenticator. The message counter is set to zero when a synchronisation message is received. A synchronisation message is also sent at startup or after a reset of the FVM. In such a case the trip counter stored in NVM gets incremented and the reset counter is reset to 0.

2.2. Reconstruction of the Freshness Value

When transmitting only a truncated FV along with each PDU, situations can occur in which the receiver will not be able to correctly verify the authenticity of the received I-PDUs due to a mismatch of the current FV, for example after an ECU reset by a watchdog timer or woken up after a long sleep. Profile 3 describes in [1, Figure 6-7] how to reconstruct the counter values when only truncated FVs are transmitted and the internally stored counters do not result in a successful verification of the Authentic I-PDU. *(1)* The receiver then tries to verify the I-PDU with its internally stored FV+1. *(2)* If this verification fails, it updates the FV with the truncated value from the received I-PDU, e. g., lower 2 bits of the reset and message counters. *(3)* The internally stored value consisting of the trip counter, reset counter and upper part of the message counter is incremented by 1 and the

verification is retried. An internal counter, called *attempts*, is incremented with every repetition of *(3)* until the allowed maximum (parameter R can be configured) is reached and the I-PDU will be dropped.

The rest of this section describes two situations where it is not possible to reconstruct the correct counter value resulting in the ECU being unable to verify the authenticity of the I-PDU. As the truncated FV contains a few bits of the reset and message counters, the receiver is able to correctly verify the authenticity of I-PDUs even if it has missed some I-PDUs. We assume in the rest of this analysis that the *ResetCycle* is harmonised with the message counter, meaning that a synchronisation message is always sent when the message counter overflows. The abbreviations used in the following analysis are as follows:

R	The maximum number of verification retries.
m	The length of the message counter.
m_l	The length of the lower part of the message counter.
r_f	The length of the reset flag.
C_{last}	The message counter value of the last successfully verified message.
T	The period in which messages are sent.

2.2.1. Reconstruction of message counter fails. The first situation in which the receiver is not able to correctly verify the authenticity of an I-PDU is when it misses too many I-PDUs to be able to correctly reconstruct the actual message counter assuming that the reset counter did not change. In this situation, it is possible to reconstruct the message counter as long as no more than $2^{m_l} + R \cdot 2^{m_l}$ PDUs are missed. Overall, the SecOC module is able to recover the correct FV as long as it receives an I-PDU with a message counter C in the range:

$$C_{last} < C \leq min\{C_{last} + 2^{m_l} + R \cdot 2^{m_l}, 2^m - 1\} \quad (1)$$

The total waiting time for a synchronisation message since the last successfully verified I-PDU is

$$(2^m - 1 - C_{last}) \cdot T \quad (2)$$

EXAMPLE. We analyse a system with an 8 bit message counter $m = 8bits$; $m_l = 2bits$; $C_{last} = 0$; $R = 2$. AUTOSAR recommends a truncated FV of length 4, which corresponds to $m_l = 2bits$ and $r_f = 2bits$ in our example. We chose to illustrate the worst case scenario, when the last correctly verified counter value is 0, for illustration purposes, however, the identified interval applies for all counter values. In this scenario the receiver can reconstruct the correct FV as long as it receives one of the next consecutive 12 I-PDUs ($2^2 + 2 \cdot 2^2$, see Eqn. 1). If it misses more than 12 PDUs, respectively the ECU sleeps longer than 240 ms with a message frequency of 50Hz, the module is unable to recover and has to wait in total $2^8 - 1 - 0 = 255$ PDUs (see Eqn. 2) since the last successfully verified PDU to receive the next synchronisation message. Figure 2 illustrates this example when the SecOC module receives only the truncated FV and MAC with each PDU. The current counter values in each step are aligned with their corresponding counter and presented in binary format.

In case that the SecOC module receives a truncated FV (see Figure 1b) of $0b0000$ and the last received message counter was 0, it is still able to correctly verify the authenticity of the I-PDU when the real message counter value is $0b00001100$. Note that the module will drop the I-PDU as soon as the MAC verification fails and the number of maximum attempts R is reached. This example shows that it is not possible (with $m_l = 2bits$) to correctly verify I-PDUs with a message counter $\geqslant 0b00001101$ as more than two attempts to reconstruct the correct message counter ($R > 2$) would be needed.

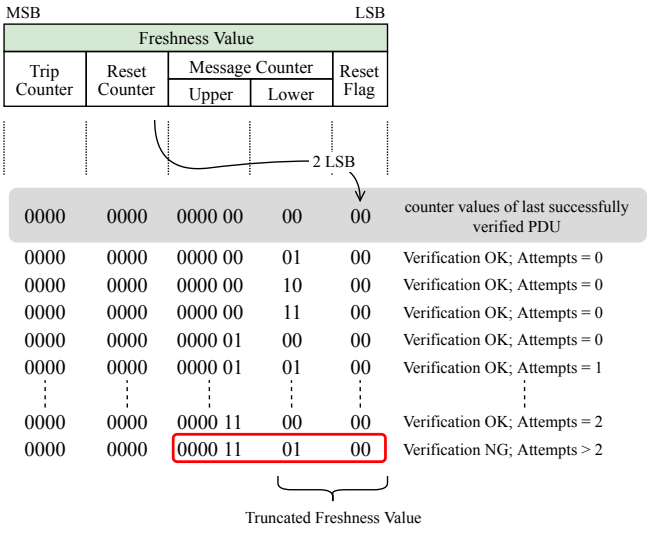


Figure 2. Example illustrating when an ECU is not able to correctly verify an I-PDU.

2.2.2. Reconstruction of the reset counter fails. The second situation occurs when the SecOC module has been inactive for a longer period and consequently missed at least one synchronisation message. In this situation, the module is able to recover the correct FV as long as the message counter is in the range described in Section 2.2.1 and as long as the truncated reset counter, the reset flag, can be used to correctly restore the FV. For example, a 2-bit reset flag can be used to correct $2^2 - 1$ reset counter increments. Overall, the interval in which it is possible to correctly reconstruct the FV with changing reset counter is illustrated in Figure 3.

3. Design Considerations and Limitations

In addition to the time span ECUs may be out of sync, non-functioning and waiting for a synchronisation message, we have identified other potential challenges when introducing SecOC Profile 3.

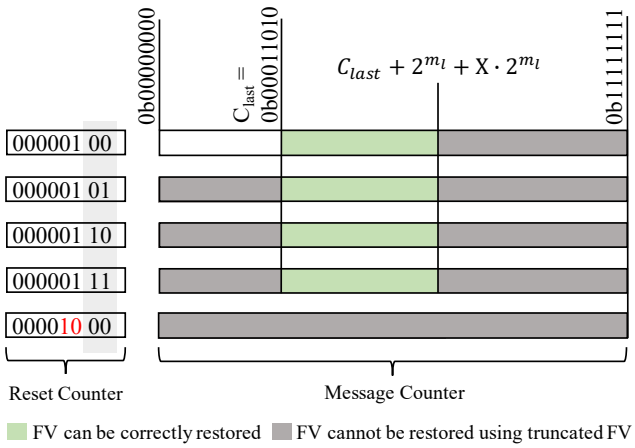


Figure 3. Example showing the cases when the ECU is able to correctly verify the I-PDU when the reset counter increases.

Freshness Value Manager (FVM). The FVM may be implemented decentralised, meaning that each sending ECU has its own instance of the FVM, or centralised, where there is one FVM for all senders and receivers.

The trip and reset counters are shared between all senders and receivers when applying the centralised approach, however, the trip counter is the only counter that has to be stored in NVM of the FVM. Storing and accessing only a small portion of the FV in NVM is faster and increases also the lifetime, which is expected to be at least 10 years, as less space in memory pages has to be written. The reset and message counters are stored in volatile memory where the message counter is the only counter maintained individually per I-PDU/message type by the sender and receivers.

CAN uses so-called CAN-IDs to distinguish between different message types. For this reason, the synchronisation message of each FVM requires its own distinct CAN-ID. Thus, a centralised approach also reduces the number of required CAN-IDs due to having one synchronisation message for all senders and receivers. The decentralised approach on the other hand requires each sender to maintain the current trip counter in their NVM as this counter is independent of the counters used by other senders.

Having a centralised FVM might bring in challenges regarding the propagation of synchronisation messages, as gateways cause additional delays. This, however, depends on the chosen placement of the FVMs, e. g., having one FVM per network segment or one for the entire internal network of the vehicle.

A centralised FVM also introduces the FVM as a single point of failure and thus increases the complexity drastically when combined with safety critical functions. As an example, a safety critical system, which is classified as ASIL D, in combination with Profile 3 would require a redundant FVM with dissimilar software and hardware redundancy according to ISO 26262 [3], the functional safety standard for road vehicles. From this perspective, a decentralised FVM is desirable for safety critical functions.

Single-I-PDU configuration. Sending data and its authenticator in a single I-PDU is faster compared to sending them separately for the reason that the data has to be kept in Random Access Memory (RAM) until the second PDU containing the corresponding authenticator is received. Heavy duty vehicles have to comply to SAE J1939 [9], which may require to send the authenticator as a separate frame for the reason that this standard defines the content of certain CAN frames to provide interoperability between a variety of equipment for heavy duty vehicles. Thus, sending data and authenticator separately allows increased security for core functions developed by the vehicle OEM while third-party modules are still able to receive the predefined messages defined in [9]. Sending two frames also allows the transmission of longer truncated values when considering the maximum payload size of 64 bits in CAN.

Complex Structure of the FV. Profile 3 introduces a FV consisting of three counters (see Figure 1a). The trip counter is incremented in units of trips, which requires a global understanding of the unit as it strongly depends on the function of the ECU. AUTOSAR specifies that the trip counter shall be incremented when the ECU running the FVM starts, on wakeup, on reset and when the power status changes from off to on. ECUs in a heavy duty vehicle might be active for several days, e. g., interior light control, whereas others will boot when the ignition is switched on. Thus, it may happen that ECUs miss the increment of the trip counter and need to wait for the next synchronisation message. Moreover, the use of the reset counter is solely to indicate that a new ResetCycle has started. For this reason, we do not see an advantage of having the reset counter separated from the message counter.

Reset Counter Overflow. There are no means to increase the trip counter when the reset counter overflows. Instead, a synchronisation message with the maximum value of the reset counter will be sent, meaning that the freshness property of transmitted messages no longer applies.

Determining the Reset Cycle. The parameter *ResetCycle* is used to define the frequency of synchronisation messages. The *ResetCycle* should be harmonised with the maximum value of the message counter in order to achieve the highest utilisation of the counter space as synchronisation messages reset the message counter.

A centralised FVM introduces additional challenges, as the *ResetCycle* is defined globally for all I-PDUs in the vehicle. In this case the counter space as well as the periodicity of the synchronisation message cannot be efficiently adjusted for individual I-PDU types.

Periodicity of Synchronisation Messages. Since synchronisation messages are broadcast periodically as defined by the parameter *ResetCycle*, receiving ECUs will be in a non-functional state until a synchronisation message is received when they are out of sync. There is no way for a receiver to notify the FVM that it needs synchronisation of the FV.

Having means to request synchronisation messages is necessary in order to provide a fast resynchronisation, such as when starting a vehicle by turning on the ignition or when

an ECU encounters a watchdog timer reset. Most vehicles, including heavy duty vehicles, have different modes of operation, such as run, accessory, parked, living and crank. *RUN* indicates that the vehicle is driving and fully operational, *LIVING* that ECUs related to the driving functionality are shut off and only comfort ECUs, such as interior lighting and infotainment, are active. Given these dynamics of the vehicle modes, it is necessary to have a fast recovery when an ECU loses track of the current FV.

From a safety perspective it is desirable to have a deterministic, maximum specified interval for synchronisation messages. In addition, having fast resynchronisation of non-safety critical functions may be important for comfort functions to provide a "premium" feeling when driving a vehicle.

4. Proposed SecOC Profile 4

We address the identified issues in our proposed profile which offers:

- a faster synchronisation of the FV.
- alignment with ISO 26262.
- less bandwidth usage due to the reduced number of control messages needed to synchronise the FV.
- a simplified structure of the FV.

We propose two modes of operation: *Mode 1* similar to Profile 3, however, a new structure for the FV is used; *Mode 2* a more efficient and faster approach that allows receiving ECUs to request synchronisation messages on demand instead of having to wait for the next periodic synchronisation message. Figure 4 provides an overview of the different configurations, modes and message types used. *Configuration 1* and *2* describe in which format the data and authenticator are exchanged, either in one I-PDU or separately. We propose using the same structure as defined in AUTOSAR for the following I-PDUs: Authentic, Secured and Cryptographic I-PDU. When truncating the FV we use the same approach as Profile 3 described in [1, Figure 6-7].

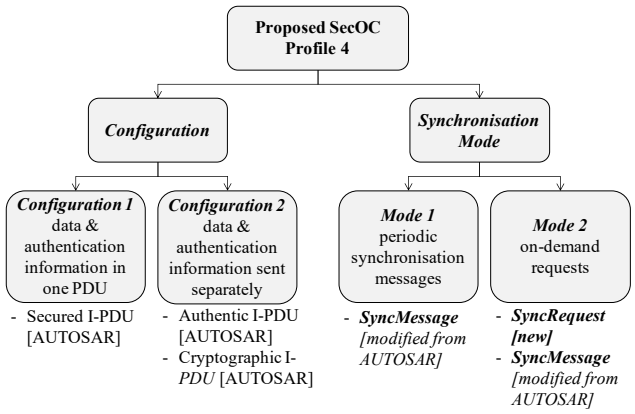


Figure 4. Overview of configuration and mode options including the involved messages.

We describe the structure of the FV and the new and modified messages in Section 4.1. Sections 4.2 and 4.3 describe the different modes of synchronisation followed by a description of default parameters and recommended values in Section 4.4.

4.1. Freshness Value and I-PDU Format

The structure of the FV and I-PDUs in our proposed method is described in this section and shown in Figure 5.

Freshness Value. Figure 5a depicts the FV, which is reduced to having only two sub-counters, i. e., a sequence counter and a message counter, which is, similar to Profile 3, split into an upper and lower part. The sequence counter is maintained by the FVM and thus needs to be stored in NVM. The scope of the sequence counter can be defined to be either one per FVM or one counter per message type. Moreover, the sequence counter may be increased due to a SyncMessage being sent for the reason that the FVM has been restarted, encountered an error, received an internal trigger to increase the sequence counter, or has received a SyncRequest.

Authentic and Secured I-PDU. The structure is similar to Profile 3 (see Figure 1b). Configuration 1 sends the Authentic I-PDU separately or combined with the authenticator when Configuration 2 is enabled.

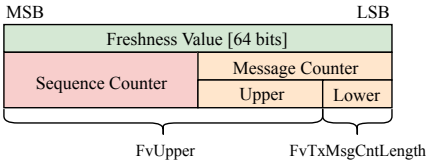
Cryptographic I-PDU. Is sent when Configuration 2 is enabled. Figure 5c explains how to combine the sequence and message counter when truncated FVs are transmitted.

SyncMessage. The synchronisation message contains the sequence counter and its corresponding MAC. As this message is used to synchronise the FV between senders and receivers, there is, similar to SecOC Profile 3, no possibility to provide freshness for this message. The nodes, however, must verify that the received sequence counter is larger than sequence counters previously used to prevent replays of old messages from an adversary.

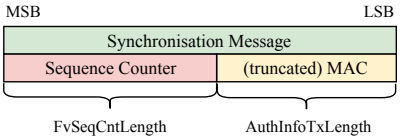
SyncRequest. Receivers may send on-demand requests for synchronisation in case they have no knowledge about the current sequence counter or cannot successfully verify the received I-PDUs. The Error-Code depicted in Figure 5d can be used to signal the FVM why the ECU demands a new SyncMessage, for instance due to consecutive verification fails or a reboot.

4.2. Sending Periodic SyncMessages (Mode 1)

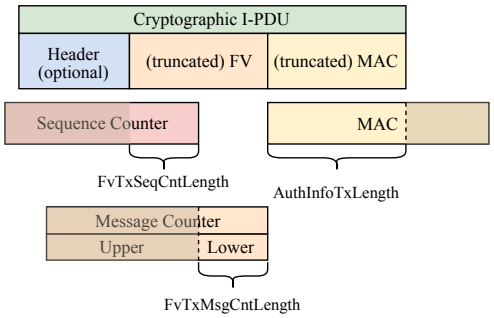
This configuration is similar to Profile 3, the parameter *ResetCycle* defines the frequency in which SyncMessages are broadcast and shall be chosen by considering the maximum time an ECU is allowed to be out of sync without degrading its functionality. We recommend using this configuration only when necessary due to its impact on the bandwidth of the underlying network.



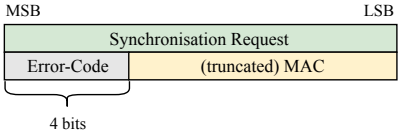
(a) Structure of Freshness Value (*Mode 1, 2*).



(b) Synchronisation Message sent either periodically (*Mode 1*) or by request (*Mode 2*).



(c) Cryptographic I-PDU sent along with Authentic I-PDU (*Configuration 2*).



(d) Synchronisation Request (*Mode 2*).

Figure 5. Structure of the proposed FV and I-PDUs.

4.3. On-Demand Request for Synchronisation (Mode 2)

This mode enables the FVM to send SyncMessages when SyncRequests are received. In situations, such as when multiple receivers wake up, i. e., changing to an active state or when an ECU being stuck in a bootloop, may occur and result in several SyncMessages being sent within a short time frame. Given these circumstances, the FVM is required to handle multiple SyncRequests within a short period of time. A dynamic parameter, such as *SyncMessageSuspend*, can be used to define the time during which SyncRequests are ignored after a SyncMessage has already been sent. Therefore situations when a SyncMessage has already been sent by the FVM but not yet received by the ECU, which sends another SyncRequest when it realises that it is out of sync, are covered as well.

SyncRequests are, similar to SyncMessages, vulnerable to replay attacks as these two messages are used to regain synchronisation of the FV. Restricting the number of SyncRequests in combination with a monotonically increasing FV, however, provide also security against attackers who inject previously recorded SyncRequests in order to overflow the sequence counter and thus force the FVM to reuse counter values. Considering the scenario of a 28 bit sequence counter and a limit of 2 SyncRequests per second would take an attacker 4.2 years until the FV is repeated.

A CAN specific implementation of the SyncRequest may make use of so-called Remote Frames (RFs) [4]. RFs can be used to request data by sending a frame with the same CAN-ID as the requested data, where only the Remote Transmission Request (RTR) bit changes. This specific solution for CAN has the advantage that no additional CAN identifier for the SyncRequest is needed and thus does not additionally exhaust the pool of available CAN-IDs. The CAN-ID is also used to prioritise the frames – the lowest CAN-ID has the highest priority (see arbitration in [4]). Coupling the priority with the CAN-ID demands further consideration when choosing an ID, as SyncRequests are event-triggered and not periodic.

4.4. Recommendations and Default Values

We recommend combining both modes, i. e., sending periodic SyncMessages and allowing ECUs to request them (Mode 1+2), for messages that need be authenticated. By enabling receivers to request SyncMessages, a larger *ResetCycle* and thus longer message counter, which in turn reduces the load on the network, can be used. The *ResetCycle* not only depends on the chosen mode, it also strongly depends on the requirements and for how long an ECU is allowed to be unable to correctly verify the authenticity of I-PDUs due to being out of sync.

The size of the FV and the authenticator depend on several factors, such as computational and storage limitations of senders and receivers, the bus technology, and current network load. Unless compliance to standards, i. e., SAE J1939 for heavy duty vehicles, is required, it has to be decided whether the data of an existing CAN frame can be split in two frames to create space for the authenticator and the truncated FV (configuration 1 in Figure 4) or if a second frame containing only the authenticator and the truncated FV (configuration 2) should be used.

Table 1 lists our recommended parameter values, described in Figures 4 and 5, for mode 1 and 2. These values are based on the AUTOSAR SecOC Profile 3 recommendations and should be considered as a base for further adaptation depending on the requirements. The number of additional attempts for reconstructing the FV depends strongly on the message frequency of the authenticated data as well as the computational power of the ECU.

TABLE 1. RECOMMENDED PARAMETERS

Parameter	Mode 1	Mode 1+2
ResetCycle	5 Hz	1 Hz
	Freshness Value	
Attempts	2	
FvLength	64 bits	
FvUpper	FvLength – FvMsgCntLower	
FvSeqCntLength	28 bits	
FvMsgCntLength	FvLength – FvSeqCntLength	
	Configuration 1	**Configuration 2**
AuthInfoTxLength	28 bits	44 bits
FvTxSeqCntLength	2 bits	4 bits
FvTxMsgCntLength	2 bits	16 bits

5. Experiments and Evaluation

The experimental setup to validate the functionality of our proposed solution is shown in Figure 6 and consists of three nodes, one sender and two receivers, which were implemented on Freescale MPC 5646C microcontrollers with a compliant AUTOSAR software using the AUTOSAR 4.3 crypto stack. The FVM is executed on the sending ECU, as we believe that a decentralised approach is more realistic to be deployed in a production environment. Furthermore, we chose to send the authentic data separated from the authentication information (i. e., use configuration 2) as this setting provides both backward compatibility and compatibility with standards specifying the frame content.

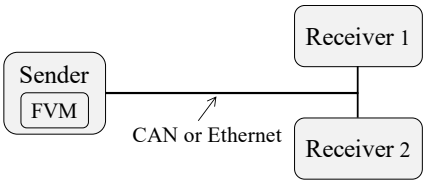


Figure 6. Experimental setup where sender and receivers either communicate via CAN or Ethernet.

TABLE 2. PARAMETERS USED FOR COMPARING MODE 1 AND MODE 1+2

Parameter	Mode 1	Mode 1+2
ResetCycle	5 Hz	1 Hz
Attempts	0	0
FvMsgCntLength	2 bits	2 bits
max. message counter value	10	100

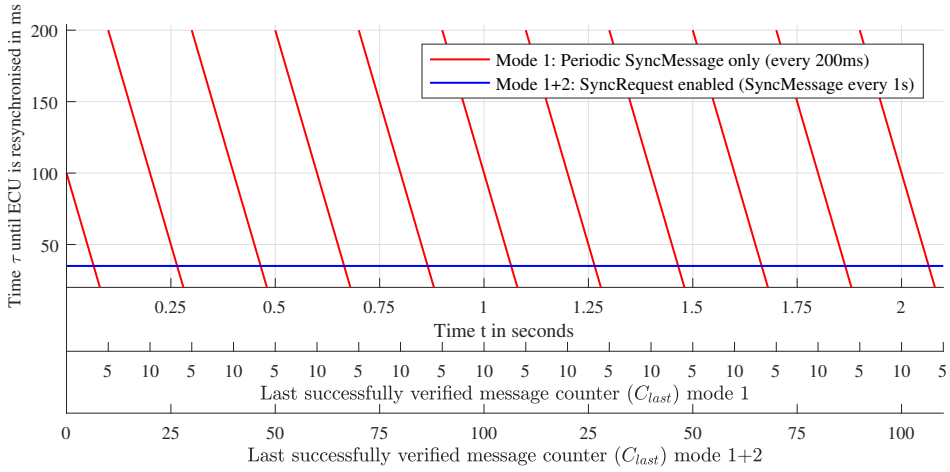
We have analysed the time a resynchronisation takes when only mode 1 or both modes were activated using the parameters shown in Table 2. These parameters where chosen to highlight the differences between the two modes, other parameters such as the length of the authenticator, are not relevant for the following analysis. Figure 7 shows the calculated times from a scenario where the receiver gets interrupted for 90 ms, e. g., due to a watchdog timer reset or an erroneous event. It shows the time τ, which is the time measured between receiving the first I-PDU after the interruption and the successful resynchronisation of the ECU. Figure 7a illustrates the behaviour of periodic SyncMessages; τ decreases with the increasing counter value C_{last}, as the next periodic synchronisation message approaches. The large τ from $C_{last} \geq 5$ is due to the interruption of 90ms, which causes the receiver to miss the periodic SyncMessage. The blue line shows that enabling requests for synchronisation messages, has a constant time τ since the generation and transmission of the SyncRequest and SyncMessage have a fixed delay. In a real setting, there will be a variation of τ depending on the bus load and priority of the messages. However, it is reasonable to assume that even without using very high priorities, these messages should always be possible to be transmitted within 20 ms. In addition, as shown in Figure 7b, enabling SyncRequests provides not only a faster resynchronisation of the FV, it also makes it possible to reduce the number of periodic SyncMessages.

5.1. CAN Test Bed

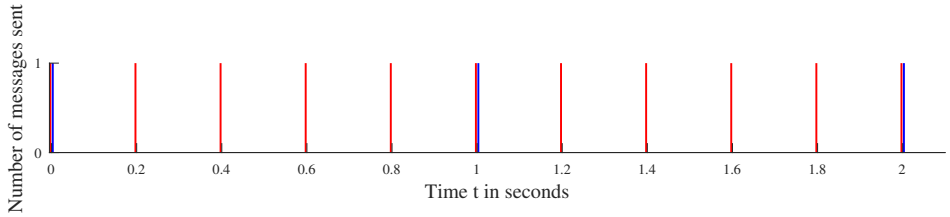
In this setting, we first analyse the processing times and delays of the sending ECU respectively the FVM. The sender is configured to send a SyncMessage every 200 ms and to accept SyncRequests. These settings were chosen to highlight that the proposed method is also faster when combined with shorter periods for sending SyncMessages. Figure 8 illustrates the frequencies of different messages transmitted on the CAN bus generating in total a bus load of 14%, where messages in cyan colour are generated background traffic. The messages with IDs 8300311 and 8300313 are the Authentic and Cryptographic I-PDU destined for Receiver 1 with a message frequency of 100 Hz. SyncMessage and SyncRequest were chosen to have a lower priority (in CAN the lowest ID has the highest priority) than the actual secured message to show that our approach also works well even when messages with higher priorities are transmitted on the network. We force the receiving ECU to get out of sync by triggering the sending ECU/FVM to immediately increase the sequence counter by 100.

In our measurements it takes the sender 17.8 ms from receiving the SyncRequest at second 5.00 to sending a SyncMessage, which corresponds to the spike shown in the chart presenting the time between SyncMessages in Figure 8.

Overall, it takes the FVM, respectively the sending ECU, in average 18.1 ms to transmit the SyncRequest, process the request and transmit the corresponding SyncMessage. Receiver 1 needed in average 25 ms from recognising that it is out of sync until having a synchronised FV again. Comparing this fast resynchronisation to sending only periodic SyncMessages as shown in Figure 7a highlights that not just



(a) Waiting time for the next SyncMessage when the ECU is interrupted for 90ms.



(b) SyncMessages being sent.

Figure 7. Analysis on the waiting time when an ECU is interrupted for 90 ms.

the waiting time can be greatly reduced, but also the period in which SyncMessages are sent can be extended to reduce the bus load of the network. To achieve the same average for resynchronisation as we have achieved by enabling SyncRequests would require sending SyncMessages with a frequency of at least 20 Hz.

Other factors such as the bus load and number of receiving ECUs have an impact on the time a resynchronisation takes when on-demand SyncRequests are enabled. In most cases it can be assumed that SyncRequests and SyncMessages can be sent within the next 20 ms on CAN even with high bus loads when assigning the priorities properly. Measures to prevent flooding of SyncMessages by many ECUs sending SyncRequests close after each other may impact the response time of the FVM as well. Such measures can be implemented on the FVM and may be to limit the number of SyncRequests within one SyncMessage period (ResetCycle) or to only send one SyncMessage within a certain time frame.

A case that might require a special handling of SyncRequests is the KeyOn event and other events where many ECUs are expected to startup simultaneously. In such scenarios a fast ECU might request and receive a SyncMessage while others are still booting. One approach is to allow more SyncRequests within one ResetCycle during startup to allow a fast synchronisation. Another approach to cope with many ECUs being out of sync is to temporarily set a faster period for sending SyncMessages and increase the ResetCycle later on. For instance, for the KeyOn scenario one may set the ResetCycle to 50 ms for two seconds and afterwards increase it to the regular period of 1 second.

5.2. Ethernet Test Bed

The test bed with the nodes communicating over Ethernet with each other was used to validate that our implementation, specifically the use of our structure of the FV, works on Ethernet as well. We chose to send the complete FV along with every message for the reason that the requirement of having a highly limited bandwidth, as in CAN, does not apply. Thus, there is no need for sending synchronisation messages or requests. We successfully validated the functionality of our proposed approach using the structure of the UDP payload shown in Figure 9.

6. Related Work

Zou et al. [11] identify the challenges of time and counter-based solutions for CAN. One of the identified challenges for counter-based solutions is the need for ECUs to be able to request synchronisation messages, however, the authors do not provide more details.

Gürgens and Zelle [2] present a CAN specific hardware-based solution to provide counter-based freshness. This

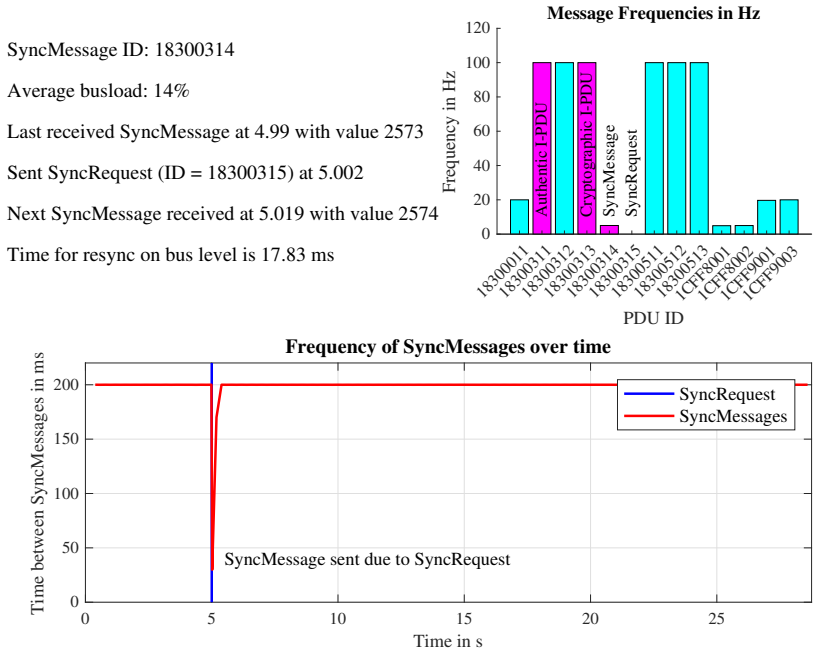


Figure 8. Results when the receiving ECU gets out of sync and sends a SyncRequest.

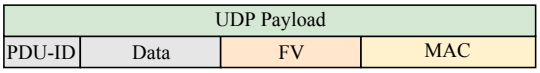


Figure 9. Structure of the UDP payload sent over Ethernet (Secured I-PDU).

method uses one counter per CAN bus where the counter gets incremented by 1 when a message is sent. The authors also mention that their proposed solution cannot be implemented in currently available ECUs as the CAN transceivers require additional functionality. Limiting the scope of the counter/FV per CAN network segment additionally increases also the delay through gateways due to the verification and subsequent generation of a new MAC using the counter of the network the message is forwarded to.

VulCAN [10] proposes a trusted computing design for message authentication including software component attestation, but leaves the resynchronisation of the FV to the underlying protocol being used.

Existing CAN authentication solutions based on industrial criteria have been evaluated by Nowdehi et al. [7]. According to Nowdehi et al., VatiCAN [8] fulfils the requirements of cost effectiveness, backward compatibility, and repair and maintenance. VatiCAN provides freshness using a nonce, but it requires to be synchronised periodically (i. e., the authors suggest every 50 ms).

7. Conclusion

Freshness is an important security property that ensures that authenticated messages have not been replayed by a malicious entity. Counter-based solutions are especially interesting in the automotive domain, as sensors and other Electronic Control Units (ECUs) rarely have a globally synchronised clock. AUTOSAR proposed two counter-based solutions, one using a single counter and another one providing additionally a master/slave synchronisation of the Freshness Value (FV).

In this paper, we focus on the second solution presented by AUTOSAR, namely SecOC Profile 3 or JASPAR. We first provide a detailed analysis on Profile 3 which shows in which situations the recovery mechanism of the FV succeeds when only a truncated FV is transmitted. Second, we study the limitations, safety impact and other design considerations when implementing a counter-based solution to provide freshness for signals in the in-vehicle network. Third, we propose an extension of Profile 3 that allows a faster synchronisation of the FV in case senders or receivers have lost track of the current FV due to, for instance, an unexpected reset, internal state change or waking up from sleep. We further evaluate our proposed solution on two test beds each communicating via CAN bus and Ethernet. The experiment shows that our proposed solution is significantly faster in synchronising ECUs. Moreover, the number of necessary control messages used to synchronise is reduced notably and will have a positive effect on the bus load.

Acknowledgement. This research was supported by the HoliSec project (2015-06894) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

References

[1] AUTOSAR 4.4.0. *Specification of Secure Onboard Communication*, 2018.

[2] Sigrid Gürgens and Daniel Zelle. A hardware based solution for freshness of secure onboard communication in vehicles. In Sokratis K. Katsikas, Frédéric Cuppens, Nora Cuppens, Costas Lambrinoudakis, Annie Antón, Stefanos Gritzalis, John Mylopoulos, and Christos Kalloniatis, editors, *Computer Security*, pages 53–68, Cham, 2019. Springer International Publishing.

[3] ISO 26262:2011 Road Vehicles – Functional Safety. Standard, International Organization for Standardization (ISO), 2011.

[4] ISO 11898-1:2015 Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling. Standard, International Organization for Standardization (ISO), 2015.

[5] Tetsu Iwata, Junhyuk Song, Jicheol Lee, and Radha Poovendran. The AES-CMAC Algorithm. RFC 4493, June 2006.

[6] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015, 2015.

[7] Nasser Nowdehi, Aljoscha Lautenbach, and Tomas Olovsson. In-vehicle CAN message authentication: An evaluation based on industrial criteria. In *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017.

[8] Stefan Nürnberger and Christian Rossow. – vatican – vetted, authenticated can bus. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 106–124, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[9] Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document. Technical report, SAE International, 08 2013.

[10] Jo Van Bulck, Jan Tobias Mühlberg, and Frank Piessens. Vulcan: Efficient component authentication and software isolation for automotive control networks. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC 2017, pages 225–237, New York, NY, USA, 2017. ACM.

[11] Qingwu Zou, Wai Keung Chan, Kok Cheng Gui, Qi Chen, Klaus Scheibert, Laurent Heidt, and Eric Seow. The study of secure can communication for automotive applications. In *WCX 17: SAE World Congress Experience*. SAE International, mar 2017.