MaCAN protocol specification

Michal Sojka, Ondřej Kulatý
Czech Technical University in Prague
sojkam1@fel.cvut.cz

Version 0.0-draft
March 26, 2015

Contents

MaCAN (Message authenticated CAN) is a cryptographic protocol proposed by O. Hartkopp et al. [3] to increase security on the CAN bus. This document provides a detailed description of this protocol.

1 Introduction

MaCAN protocol was created in order to increase security of on-board communication in cars. Historically, it was assumed, that attackers do not have the access to the on-board networks, but it has been shown several times, that in modern cars, with several remotely accessible interfaces, this assumption does no longer hold. Traditionally, on-board networks were designed for reliability, robustness and deterministic real-time operation, but they offer no security features, like message authentication. The MaCAN protocol is designed to preserve real-time capabilities of the CAN bus and to be easily implemented in real CAN environment – it is fully backward compatible with existing and deployed technology.

The basic idea is to allow an electronic control unit (ECU) to send an authenticated message within a single CAN frame, which can be verified by either a single receiving ECU or by a group of receiving ECU. Either permanent signal authentication or on demand signal authentication is possible. MaCAN provides the following security properties: authenticity, integrity and to some extent freshness.

1.1 Restrictions on CAN bus

Several restrictions apply when trying to add a message authentication capabilities to the CAN bus, due to it's properties.

Message length Standard CAN frame has maximum payload length of 8 bytes. In order to send an authenticated 32 bit signal, only 4 bytes remain for the signature. Although CAN FD offers payload length up to 64 bytes, MaCAN focuses on the standard CAN specification. The reasons are that CAN FD is not as widespread as the standard CAN is.

Available resources The number of CAN identifiers as well as the bus bandwidth is limited. Use of a bidirectional protocols based on challenge-response authentication on the CAN bus would require $m \times n$ CAN-IDs, where m is the number of nodes (ECUs) and n the number of messages. As number of CAN-IDs is limited, this is a significant restriction for protocol designers. Bidirectional protocols also bring significant overhead in terms of used bandwidth and this would limit the real-time capabilities of the bus.

1.2 Requirements

The following requirements apply for a security protocol to meet the restrictions outlined above.

Message authentication As the number of ECUs responsible for critical functions in today's cars is increasing, it is necessary to be able to verify the origin of the messages. That means ECUs should not react on forged messages sent by an attacker. Since most messages are highly time dependent, the concept should also provide mechanism which will prevent an attacker from replaying intercepted frames and guarantee freshness of the authenticated messages.

Real-time capabilities A car network has strong real-time requirements, i.e. signals need to be delivered within their hard deadlines. The security concept should preserve this real-time properties of the bus. Some ECUs need to respond within milliseconds. If the signal signature would be spread across multiple messages, it is not guaranteed that the ECU would be able to verify origin of the signal within this time frame.

Flexibility It should be possible to apply a message authentication to certain signals, while others may need no protection. Additionally, it should be possible to authenticate individual signals on demand, while preserving their real-time properties.

Backward compatibility The concept should allow concurrent use of non-security equipped nodes with those able to authenticate messages. This is important for implementation in existing vehicular networks. This means that signals that were previously defined on the network should be accessible to all nodes, even after an authentication element was added to them.

1.3 Concepts

Due to the limited size of the payload in the CAN frame, *message authentication codes* (MACs) are used to sign messages. To better utilize hardware implementation, CMAC mode is used. This mode uses a block cipher as a cryptographic element in MAC. It is based on similar idea as symmetric cryptography, therefore all nodes willing to communicate together must share the same group key. MaCAN allows authenticated communication between two nodes as well as between a group of nodes. In the group however, all nodes must agree on common trust level, since all share the same group key.

1.3.1 MAC truncation

Due to limitations of the maximum payload in the CAN frame and the need to preserve real-time capabilities, MAC signature is truncated to 32 bits. This makes it easier to guess, but the number of guesses an attacker can perform is limited by a relatively low speed of the CAN bus. Short lived session keys are used to additionally increase security level.

CMAC length depends on the block size of the used cipher. In case of AES, a quick block cipher widely implemented in the hardware, it is 128 bits. Level of truncation depends on how many guesses can an attacker perform during the lifespan of the session key. According to [2] CMAC length should satisfy following inequality:

$$T_{\text{len}} = \log_2 \frac{MaxInvalids}{Risk} \tag{1}$$

T_{len} is the output length of CMAC and $MaxInvalids$ is the number of attempts an attacker can perform before validity of the session key expires. According to [2] the minimum length of CMAC should be 64 bits. However, thanks to the low speed of the bus and short-lived session keys, it can be truncated to 32 bits. One authentication attempt every 20 ms results in 8640000 attempts during a maximum session key lifetime of 48 hours. According to equation 1, 32 bit CMAC results in risk of 1 in 500 of guessing a single signature during the lifetime of the key, yet still it is not guaranteed, that this forged message will be accepted. Attempts at such a high frequency can be recognized by comparing the actual and the expected message frequency on the bus and also by restricting the number of responses to false signatures.

1.3.2 Authentication process

TODO: Clarify/update this

Since CAN frames with the same CAN-ID may be used carry more signals, two different types of frames are presented. The first type is a dedicated frame containing only one signal, its ID and its signature as the payload while preserving the original CAN-ID of the frame. This can be used for a limited amount of signals, since more frames need to be send.

The second type uses a dedicated CAN-ID in addition to an unauthenticated CAN frame and contains a signal and its signature only. This type can be used for the on-demand authentication of usually unauthenticated signals.

1.3.3 Message freshness

As described above, use of bidirectional protocols as a way to protect the messages against replay attacks is not an option. Use of a counter values as a source of freshness can also be

problematic, since ECUs might be unavailable for a number of reasons. Therefore, in this concept, nodes use synchronized time to protect messages. To maintain synchronized time, a time server (TS) is added to the system. It broadcasts current timestamp T in a regular intervals and can authenticate them upon request.

2 Network configuration

MaCAN network consists of a set of nodes (ECUs) identified with an 6 bit identifier called ECU-ID. Besides application nodes, two special nodes exist: a key server and a time server.

All nodes in the MaCAN network must share common configuration. The configuration comprises the following information:

- ECU-ID of the the key server and the time server.
- TODO: Mapping between ECU-IDs and Group-IDs (see Section 4.1.1).
- Long term keys: Every node shares a symmetric long-term key (LTK) with the key server to allow safe distribution of the session keys.
- CAN identifiers
  - *time-id*, *atime-id*: CAN identifiers used in the time distribution protocol (Section 4.2).
  - Crypt frame identifiers: These are used for "crypt frames" (Section 3.1) and are derived from ECU-ID by function $f_i : \mathit{ECU\text{-}ID} \rightarrow \mathit{CAN\text{-}ID}$. Typically, f_i is implemented as a lookup table.
  - Authenticated signal identifiers: These are derived from signal number by function $f_s : \mathit{sig\#} \rightarrow \mathit{CAN\text{-}ID}$. Typically, f_s is implemented as a lookup table.
- Timing parameters
  - *time_div*: Duration of one MaCAN time unit
  - *skey_validity*: Session key validity time. No session key can be used longer than this time. Authors of the protocol recommend that the validity of the session key is 48 hours.
  - *skey_chg_timeout*: Timeout for waiting for the session key in key distribution protocol (Section 4.1).
  - *time_req_sep*: Minimum time between requests for authenticated time from one node
  - *time_delta*: Maximum time difference between local MaCAN clock and time server clock. If greater difference is detected, the node should ask for authenticated time and synchronize its local MaCAN clock based on it.

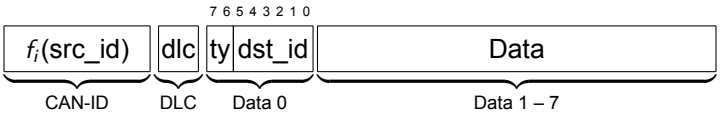
3 Message formats

This section describes the format of messages used by MaCAN. Standard (SFF) or extended (EFF) CAN frames are used to transfer these messages. Thus, each frame is identified with a CAN-ID, which is either 11 bits (SFF) or 29 bits (EFF) long and has up to eight bytes of data payload, as determined by the data length code (DLC) field.

TODO: CMAC is AES.

3.1 Crypt frames

Since the standard CAN frame does not allow direct addressing, a new type of format, called *crypt frame* introduces new partitioning scheme of the CAN frame payload as shown below.



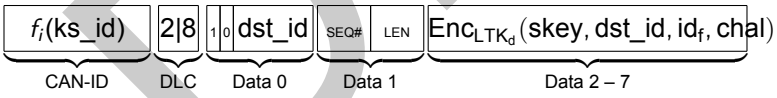
Every node uses a dedicated CAN-ID for the crypt frames it sends. This is which is derived from its ECU-ID by function f_i (see Section 2). This allows the receiving node to identify the sender's ECU-ID (src_id) from the CAN-ID in the received crypt frame. Two flag bits are mapped to the two most significant bits in the first byte in the payload and are used to distinguish the different message types in the newly introduced protocols. The remaining 6 bits are used for the destination node's ECU-ID (dst_id).

The following sections contain detailed description of different messages that use crypt frame structure.

3.1.1 CHALLENGE message

The CHALLENGE message is used for sending challenges in key distribution (4.1) and time distribution (4.2) protocols. The Challenge field contains a random number.

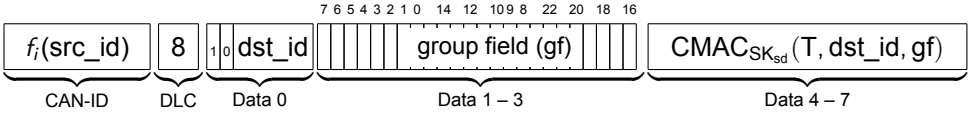
3.1.2 SESS_KEY message



The SESS_KEY message is used by the key server to send encrypted session keys to other nodes. See Section 4.1.

The plain text to be sent in the SESS_KEY messages is 24 bytes long and comprises of: 16 bytes long session key, 2 bytes of IDs of both nodes (or a node and a group) sharing the key and 6 bytes of the challenge. The AES-Wrap algorithm is used to encrypt the plain text, which will result in 32 bytes long cipher text [4]. Due to the limited length of the payload, this message cannot be sent in a single CAN frame. Therefore, the message is split into several parts and sent using format depicted above. Basically it is a crypt frame discussed earlier with added byte consisting of 4 bit *SEQ#* value, which is a sequence number of the frame and a 4 bit *MSG_LEN* value, which corresponds to the payload length. Given that the maximum payload length of this frame is 6 bytes, six frames must be sent by the key server to deliver the wrapped key to the requesting node.

3.1.3 ACK message



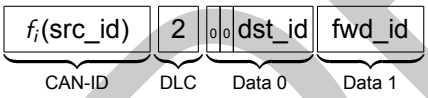
The ACK message is used by nodes nodes to notify others about having received the session key. See Section 4.1 for more details about the protocol. The 24 bit wide group field indicates which nodes the sender is aware of sharing the session key with. See Section 4.1.1.

The original paper does not specify exactly how nodes should be mapped to this field, but at least two approaches are possible:

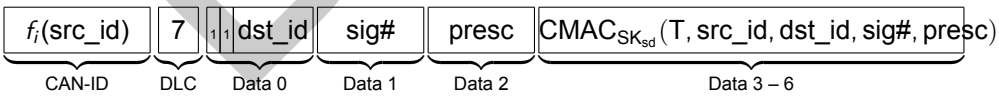
1. Each node in the communicating group is assigned to a fixed bit position in the *group_field*. This information must be statically defined at design time for every communication group and all nodes must share it.
2. Node's bit is determined by taking the numeric value of its ECU-ID. If this value is *n*, then *n*-th bit in *group_field* corresponds to that node. This is suitable only for networks with maximum of 24 nodes. Also nodes need to have their ECU-IDs in range 0 – 23.

If another ECU already shares its key, but does not find itself in the received ACK message, it transmits its own ACK message with its bit set to 1. Every time an ECU receives an authentic ACK message, it internally marks nodes found in *group_field* as ready. Once all partitioning nodes are marked as ready, secure communication channel is established.

3.1.4 REQ_CHALLENGE message

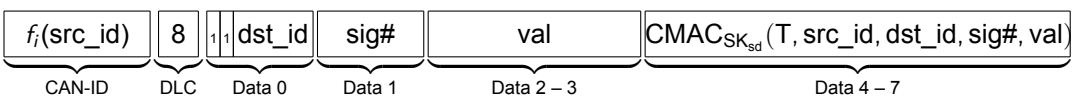


3.1.5 SIG_AUTH_REQ message



Request for authenticated signal. See Section 4.3.

3.1.6 AUTH_SIG message

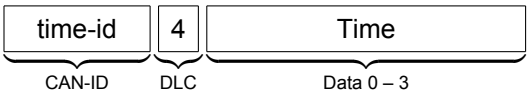


This message is used to send authenticated signals (up to 16 bit wide). For another way to send authenticated signals see Sec. 3.3.

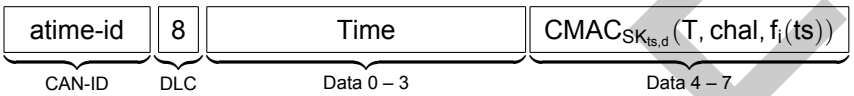
3.2 Time related messages

The following messages are used by the time server to distribute time in the MaCAN network. They use a dedicated CAN-ID time-id and atime-id. See Section 4.2 for more details.

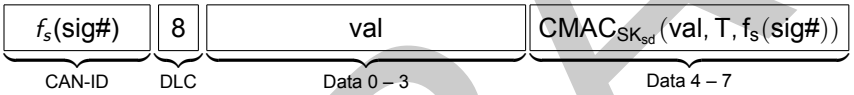
3.2.1 TIME message



3.2.2 AUTH_TIME message



3.3 AUTH_SIG32 message



This message conveys authenticated signals that are wider than 16 bits or that have real-time requirements that cannot be met by using AUTH_SIG message (Sec. 3.1.6) due to its (perhaps) low priority.

4 Protocols

This section describes the protocols used to distribute session keys, synchronize time and sign signals on request.

4.1 Key distribution protocol

To enable exchange of authenticated messages between a pair or a group of nodes, all participants must share the same session key, which is valid only for limited amount of time. The key distribution protocol is used to wsecurely distribute session keys to the nodes. A new entity called *key server* (KS) is responsible for generating and distributing the keys. The key distribution protocol is illustrated in Fig. 1. Challenge-response authentication is used to guarantee the authenticity of the key server.

The procedure starts with ECU_i sending the challenge to the key server. The key server responds with a series of SESS_KEY messages containing the session key encrypted using the long-term key. Optionally, it notifies ECU_j that a new key was generated. After decryption the key and verifying the challenge value, ECU_i send an ACK message informing that it has the new session key. Then, analogous procedure is repeated for ECU_j. When ECU_i receives the ACK from ECU_j, it verifies its authenticity and since it does not see itself in the group field, it sends another ACK

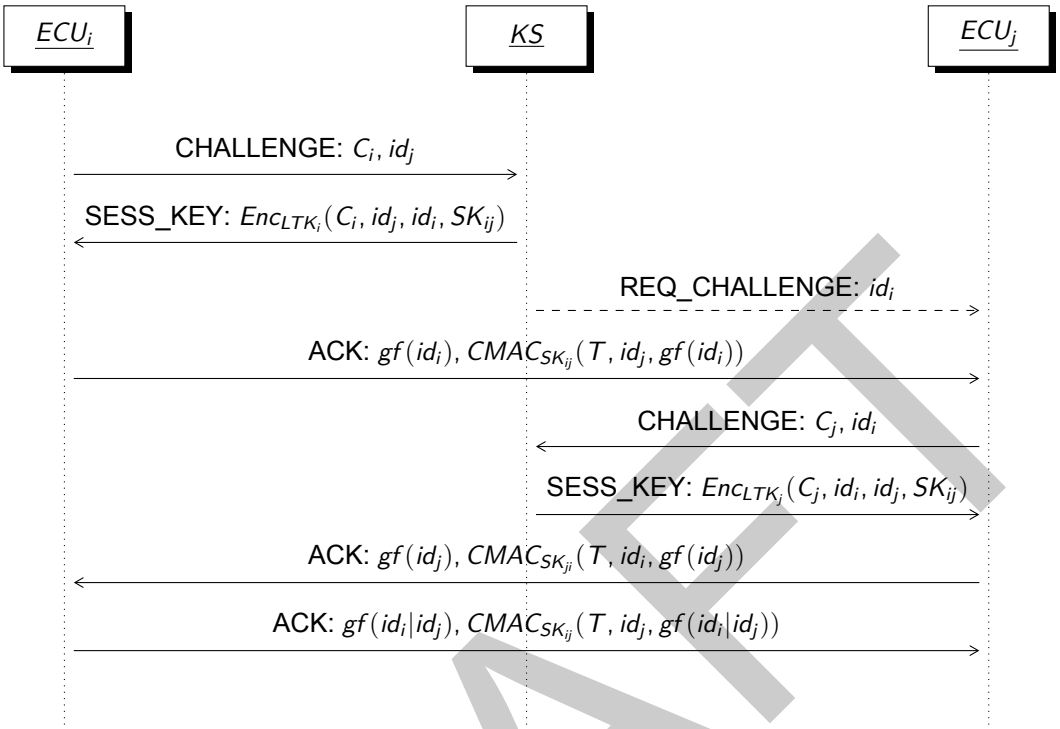


Figure 1: Session key establishment

with both id_i and id_j bits in the group field set. Note that ECU_j did not add id_i to the group field, because it could not verify the authenticity its ACK without the session key.

The key server maintains the session keys and generates a new one only if there is no previous session key, or when the current one is about to expire soon. If this is not the case, the current valid session key is sent upon key requests.

In the original MaCAN protocol specification, CMAC of all ACK messages in the session key establishment procedure had always id_j as a parameter, independently of the destination of the message. This is however a security flaw, which might lead to situation, where ECU_i believes that the session key has been established, but ECU_j has not received the session key [1]. We have modified CMAC functions in the ACK messages according to corrections proposed in [1].

4.1.1 Groups of nodes

All members of one communication group share the same session key. This can lead to problems as some node can pretend the role of any other member of the same group. Therefore authors of the original paper suggest to form the groups based on the trust level of their members. This can prevent some less trustworthy device to forge signals of some well protected ECU.

Each group is assigned a 6 bit Group-ID (similar to ECU-ID) and the key server as well as every member of the group has the list of the group members as well as their ECU-IDs. If a node requests a session key for the communication with other node, it first checks if this node is a member of the same group. If this is the case, it requests the key for the common Group-ID. As the key server is aware of the members in all groups, it will send the REQ_CHALLENGE messages to every member of this group, so all members can request the session key.

Each node in the group still uses its ECU-ID to determine the CAN-ID using f_i. Group-ID is used only in *dst_id* and *fwd_id* fields, so nodes need to filter messages on the bus addressed to their own ECU-ID or to the Group-ID.

4.2 Time distribution protocols

4.2.1 Original time distribution protocol

In order to assure freshness of the authenticated messages and to prevent replay attacks, each node is equipped with an internal counter, which maintains current time. This time is then used as one of the inputs to the CMAC function. The problem of counter inaccuracy and synchronization (e.g. after ECU restart or wake up) is solved by adding a new entity called *time server* (TS). Its purpose is to maintain and provide a reliable, precise, monotonically increasing time signal to all nodes on the network and to provide authenticated time upon request.

Time server broadcasts the current time as a 32 bit number to the bus in periodic intervals. Every node is able to receive this time signal and compare it with its internal counter. If the two values differ more than *time_delta* value, resynchronization is necessary. This can happen after ECU reboot or just because the internal timers of the ECUs drifted away from each other.

Authenticated time distribution protocol is used to request authenticated time from the time server and is illustrated in Fig. 2.

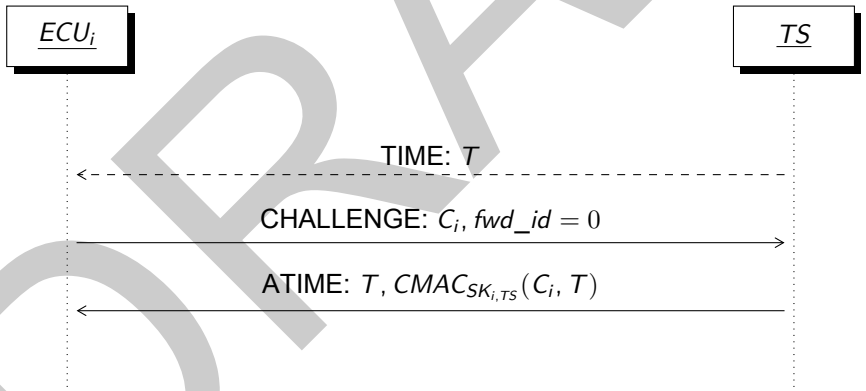


Figure 2: Time authentication protocol

First, the requesting node sends the CHALLENGE message to the time server where *fwd_id* is set to 0. Time server then replies with the AUTH_TIME message containing the last broadcasted timestamp together with the signature. Challenge-response authentication is used since we cannot use time as input for this signature to prevent replay attacks. The format of this authenticated time message is illustrated in Fig. 3.2.2. If an attacker tries to forge a plain time message to alter time of the ECU, he will not be able to reply with the authenticated time message, so ECUs are able to detect this attack and ignore attacker's message.

Group keys should not be used in an authenticated time distribution protocol, since the integrity of the CHALLENGE message cannot be verified by all members of the group. Every node must request signed time individually, using its ECU-ID, when synchronization is necessary.

If there were only one recipient of the time signal or one group with common session key, it would not be necessary to use challenge-response protocol. After the initial challenge and response step, time signal could be broadcasted together with its signature. However, if there were more

ECUs, the time server would have to broadcast n different authenticated time messages for every point in time, since every nodes requires its own signature. This would decrease precious available bandwidth on the bus, so authors decided to use the approach described above.

4.3 Signal authentication

If node ECU_i wants to receive a signal *SIG#* from node ECU_j, it must first send an authenticated signal request to that node (see Fig. 3). Structure of the signal request message is illustrated in Fig. 3.1.5. Depending on the *prescaler* field, signed messages of a periodic signals will be sent with frequency defined in Eq. 2. If the *prescaler* is set to 0, only next message will be signed. Signed messages do not replace the unsigned ones, but are rather send in addition to them.

$$f_{\text{signed}} = \frac{f_{\text{unsigned}}}{prescaler} \tag{2}$$

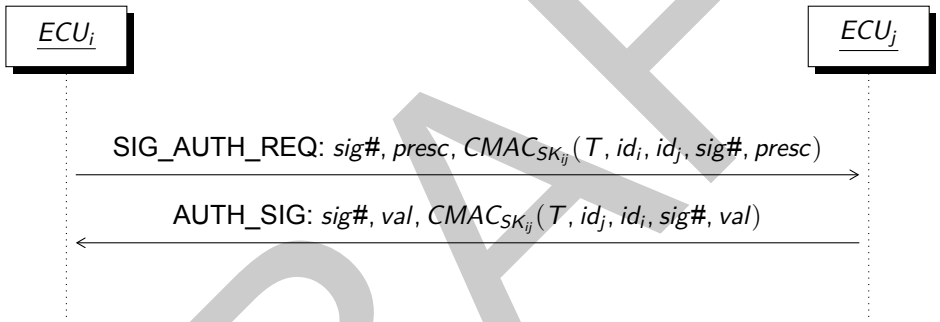


Figure 3: Signal authentication

At the design time, all signals must have assigned an 8 bit identifier, which is used to specify the requested signal in the SIG_AUTH_REQ message. Based on type of the signal and network configuration, the signed signal can be sent in two different formats: AUTH_SIG or AUTH_SIG32.

TODO: CMAC checking – the message can be delayed by the bus arbitration so CMAC must be checked also against older time stamps. How many time stamps must be tried can be derived from schedulability analysis.

References

[1] Alessandro Bruni et al. “Formal Security Analysis of the MaCAN Protocol.” In: *Proceedings of the 11th International Conference on Integrated Formal Methods, IFM 2014*. Ed. by Elvira Albert and Emil Sekerinski. Lecture Notes in Computer Science. Springer, 2014, pp. 241–255. ISBN: 978-3-319-10180-4. DOI: 10.1007/978-3-319-10181-1_15.

[2] Morris J Dworkin. “SP 800-38B. Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication”. In: (2005).

[3] Oliver Hartkopp and Roland Schilling. “MaCAN – Message Authenticated CAN”. In: *Escar Conference*. Berlin, Germany, Nov. 2012.

[4] Jim Schaad and Russ Housley. “RFC 3394, Advanced Encryption Standard (AES) Key Wrap Algorithm”. In: *Internet Engineering Task Force* (2002).