No excuses for not securing your CAN FD communication!

(Photo: Fotolia)

This is the third part of an article series about CAN security. It summarizes some of the discussions at CiA's security workshop and the inaugural meeting of CiA's Interest Group (IG) "CAN cyber security".

Did you use security on Classical CAN in the past? No? If you're like the authors, your arguments for not using security on CAN could have been along the following lines:

- It's only internal communications! Nobody other than a local technician will ever have access to it!
- There are no standards!
- The CAN frames are too short!
- My micro-controller couldn't handle it!

We did our best to refute the first argument in the first two articles of our series about CAN security [1]. This day and age, with more and more devices getting connected to the Internet either directly or indirectly, nobody should feel entirely safe anymore. Even if you still do, upcoming government regulations may force you to think otherwise.

But there are still no standards!

The CiA recently established the IG "CAN cyber security" in order to harmonize definitions and propose solutions. Because generic standards and solutions for end-to-end security beyond CAN (FD) readily exist (like TLS, security for transport layer) and can easily be tunneled through segmented CAN communications, the group excluded these from their scope of work and will instead focus on lower-level, CAN-specific solutions that are not standardized yet.

As a member of this group, we are able to present our generic, higher-layer-independent solution for a security layer – just above the CAN FD data link layer – that we also want to share with you.

Data size? Performance?

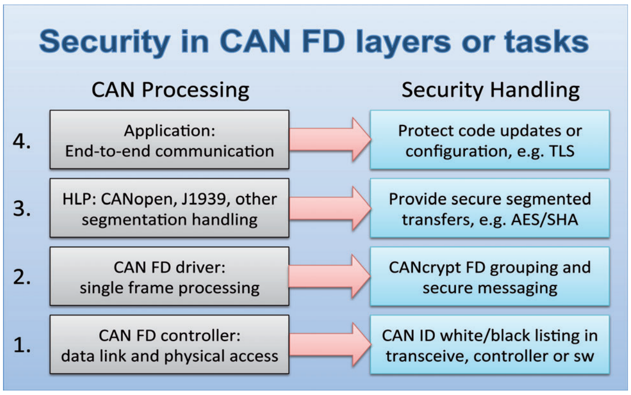
While in the past CAN with its 8-byte maximum payload per frame severely limited options to include security in each frame, with CAN FD this is no longer the case. With up to 64 data bytes in a frame, it should always be possible to make room for security data like signatures. In CAN, there are basically two use cases: "small data" and "large data". Small data, like process data in the form of sensor readings or actuator commands, is usually measured in bits and takes up a few bytes at the most. Even grouping multiple such values, you can always reserve some space in the – now much longer – CAN FD frame. Secure grouping of multiple nodes that transmit such small data is the primary concern in the interest group. Large data on the other hand, like configuration tables, logging data or firmware updates, will always need segmentation. In most cases, large data is better secured with some end-to-end method that we won't cover here or in the interest group. But if security is introduced to lower-level segmentation protocols regardless, the additional overhead for security should be easy to accept, even if it means not being able to use full frames for the payload. As a reminder, even in CANopen over Classical CAN and without security, segmentation would not start at nine bytes – one byte over the maximum eight bytes in a frame – but five, to leave room for the protocol overhead.

As for the micro-controller performance, there are no indications that semiconductor companies are going to include CAN FD controllers in very low-end micro-controllers. ▷

After all, they have to be able to handle not only eight times the data per frame, but also the faster data rate that the “FD” in CAN FD stands for. Typically, 32-bit cores with a core clock beyond 40 MHz are used with CAN FD – providing enough power to also run security algorithms. In addition, new micro-controllers in the roadmaps are planned to often have built-in hardware support for security algorithms like AES and others. We should take advantage of them whenever possible.

Where do we need security, and what kind?

Looking at possible CAN (FD) security solutions for individual layers, we can split these into four groups as illustrated in Figure 1:



Security in CAN FD layers or tasks

CAN Processing → Security Handling

4. Application: End-to-end communication → Protect code updates or configuration, e.g. TLS

3. HLP: CANopen, J1939, other segmentation handling → Provide secure segmented transfers, e.g. AES/SHA

2. CAN FD driver: single frame processing → CANcrypt FD grouping and secure messaging

1. CAN FD controller: data link and physical access → CAN ID white/black listing in transceive, controller or sw

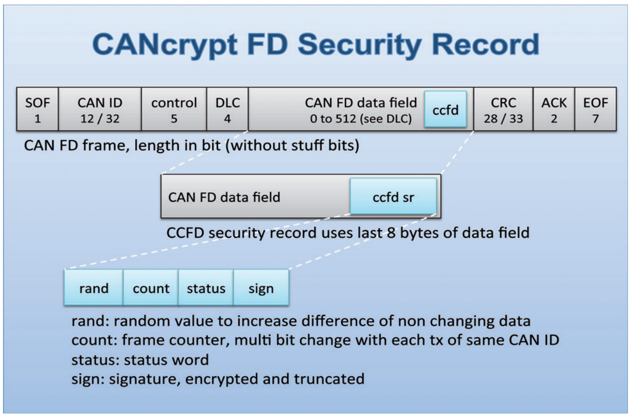
Figure 1: CAN layers and processes (Photo: EmSA)

1. Security solutions directly within the physical and data link layers: like black or white listing of CAN frame identifiers to ignore data “not authorized” for a device.
2. Security solutions for single CAN frames: the security information is embedded within the CAN frame, reducing the maximum available data size. In Classical CAN a second frame (like the CANcrypt [2] preamble message) could be used if full data size is required.
3. Security solutions for messages to be transmitted in several CAN frames: as segmentation always requires some higher layer protocol handling, such solutions should be part of the higher-layer protocol. This would secure the entire data set transported in multiple segments, and not each individual segment.
4. Security solutions for communications beyond CAN: once communications leaves the CAN bus, security mechanism “for that outside world” should be implemented. CAN, or the higher layer protocol used, only tunnels this information to the CAN device.

For the reminder of this article, we will focus on number 2, securing individual frames. We will have a closer look at the other solutions in future articles.

The security record and the digital signature

Each secure message has a security record embedded at the end of the CAN FD data field. Figure 2 shows the CAN FD frame with the location of the security record.



CANcrypt FD Security Record

CAN FD frame, length in bit (without stuff bits)

CCFD security record uses last 8 bytes of data field

rand: random value to increase difference of non changing data
count: frame counter, multi bit change with each tx of same CAN ID
status: status word
sign: signature, encrypted and truncated

Figure 2: The CANcrypt FD Security Record (Photo: EmSA)

The default security record consists of four 16-bit values:

- Random data: increases entropy and decreases predictability of content
- Frame counter: to guard against re-play attacks, increases on transmit of frame with same CAN ID, increment value is such, that four or more bits change occur per count
- Status word: padding info, current key identification
- Signature: derived from a 64 or 128-bit checksum, encrypted, truncated to 16 bits

As popular secure hash digests like SHA-256 are quite big, CANcrypt FD uses a truncated encrypted checksum as digital signature to authenticate the message. Figure 3 illustrates how it is generated (here, with a 16-bit signature and 64-bit block cipher).

First, a buffer the size of the key is initialized. Instead of an all-zero initialization, it can also be based on another shared secret. For example, if the shared secret key is larger than required (e.g. 128-bit key, but using 64-bit encryption method), then the key could be split, one half used as main key, the other half as checksum initializer.

Second step, a checksum is build. The number of bits the checksum uses is equal to the buffer width (64 or 128). It covers the CAN ID, the data field before the security record and the security record without the signature. Third step, the buffer gets encrypted using the current dynamic key. Forth step, the checksum is truncated and the least-significant 16 bits of the buffer are used as the signature.

Powering up

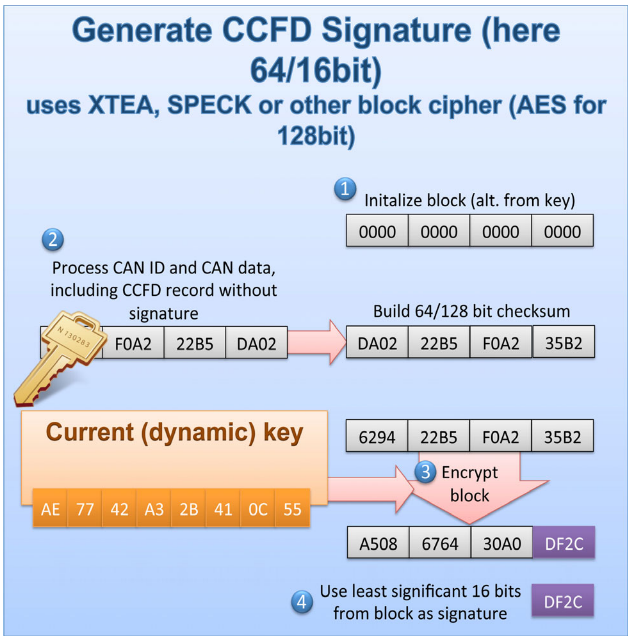
On power up, the devices actively participating in the dynamic key generation process start a secure grouping cycle, as in the Classical-CANcrypt grouping. All nodes that wish to participate in secure communications must monitor the key generation processes to maintain a local copy of the dynamic key. This cycle is illustrated in Figure 4. All participating devices exchange random values, which are used as an initialization vector for generating the next key from a previous or known key.

Once the dynamic key is generated, all devices (also those not actively participating in the key generation) may start using it by transmitting secure messages on the CAN FD network.

Known limitations

In its current state, CANcrypt-FD requires one dedicated CAN ID for each device and a default of eight bytes for the security record, added to the data field of every secure message. Future versions may use different-size security records, if required by specific applications.

The CANcrypt-FD mechanisms only provide one element for the security of CAN FD communications. Which specific keys and algorithms are used and how frequent a dynamic key is updated, is still application-specific. Here, we need further recommendations to avoid insecure configurations.



Generate CCFD Signature (here 64/16bit)
uses XTEA, SPECK or other block cipher (AES for 128bit)

1 Initalize block (alt. from key)

2 Process CAN ID and CAN data, including CCFD record without signature

Build 64/128 bit checksum

3 Encrypt block

4 Use least significant 16 bits from block as signature

Figure 3: Generating the digital signature (Photo: EmSA)

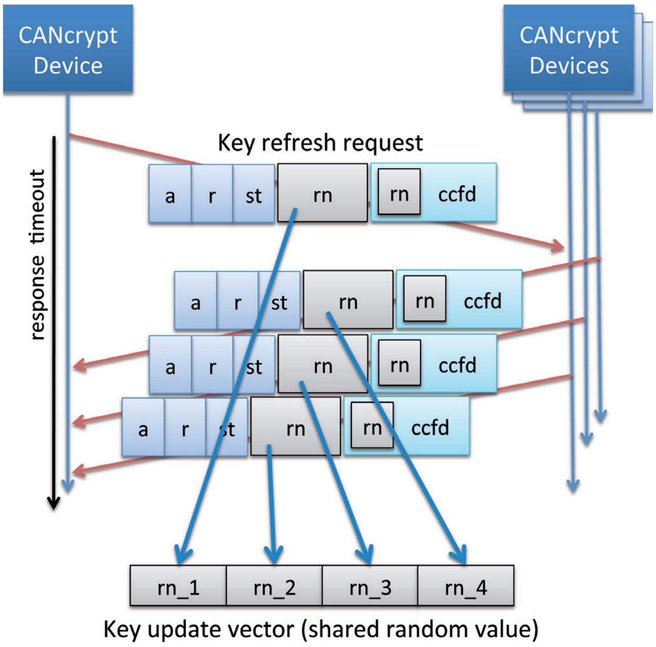


Figure 4: Generating an initialization vector for key generation (Photo: EmSA)

Brute-force or penetration testers can only be kept at bay if a reliable reset with power-up delay is implemented. If a device requests re-grouping too often or grouping fails repeatedly, then a reset with an increasing power-up timeout is required.

Outlook

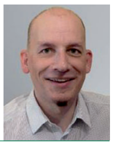
A demo of CANcrypt-FD is available for the NXP LPC54618 processor. By the time of this publication it should be available for free download from the Embedded Systems Academy web pages.

We will work closely with the CiA CAN Cyber Security group to see if this solution is suitable for a wider range of applications. It is our desire to keep the main framework and security algorithms open, so that they can easily be reviewed by all interested security experts or testers.

References

[1] Security expectations vs. limitations, Olaf Pfeiffer & Christian Keydel, CAN Newsletter 1/2018 (part 1) and CAN Newsletter 2/2018 (part 2)

[2] Book: Implementing Scalable CAN Security with CANcrypt, Olaf Pfeiffer, ISBN 978-0998745404



Authors

Olaf Pfeiffer
Christian Keydel
EmSA (Embedded Systems Academy)
info@esacademy.com
www.esacademy.de