Closing the CAN security gap: CANopen (FD) security

(Source: Adobe Stock)

Cybersecurity has stopped being a topic that CAN designers can further delay until the next product generation. Embedded Systems Academy (EmSA) provides security publications and matching reference implementations that let teams ship CAN (FD) and CANopen (FD) products with cryptographic protection already today.

The EU Cyber Resilience Act (CRA) starts to apply from late 2027 and reaches every product with a digital element. The NIS2 (Network and Information Systems) directive is being transposed into national law across Europe and obliges operators of essential and important entities to apply risk-appropriate technical measures. IEC 62443-4-2, the component requirements for industrial automation and control systems, is now routinely cited in tender documents.

Gap between regulation and CAN security solutions

At the same time, the formal CAN-side answer is still in motion. The CiA cybersecurity-related working groups are working on solutions, but standardization takes time and system designers do not have the luxury of waiting. CRA documentation and certifications are scheduled for the next year.

This is where EmSA's security publications and matching reference implementations come in. They are not a replacement for the future standards, but they let teams ship CAN and CANopen products with real, auditable cryptographic protection today, using the same threat language the regulations use.

Two tiers of security for CAN

There are two complementary security approaches. They sit at different layers of the stack and they solve different challenges.

The heavier option is SPsec (small-packet network security sublayer). SPsec is a security sublayer that sits immediately above the data link layer and has the capability to protect every CAN FD frame on the network, including the high-rate cyclic process data that dominates a running network. It can use full authenticated encryption per frame, reducing a frame's payload from 64 bytes to 48 bytes. The downside is the price of admission: SPsec is deeply integrated between the CAN driver and the upper stack and integrating it into an existing device is invasive. SPsec is documented in EmSA's SPsec101 (concept), SPsec201 (generic specification) and the network mappings are given in SPsec301 (serial point-to-point) and SPsec302 (CAN FD, covering CANopen FD).

The lighter option, and the focus of this article, is SOFA (Secure Object Fieldbus Access). SOFA lives entirely in the application layer above CANopen and is therefore easy to integrate with CANopen stacks. It uses the object dictionary (OD) entries of type DOMAIN or OCTET_STRING as its transport, which means it works unchanged on both CANopen CC (classic) and CANopen FD networks. It secures selected objects accessed through an SDO (service data object) or USDO (universal service data object) client-server relationship. It is not aimed at high-rate cyclic process data; for that traffic class, SPsec remains the right tool.

Where the keys come from

Any cryptographic protection on the network is only as good as the keys underneath it. White paper EmSA-WP-104, Key Provisioning for Minimal Fieldbus Systems, defines a symmetric-key provisioning model for constrained CAN CC and CANopen CC devices that cannot run asymmetric cryptography at production rates. It covers the full lifecycle from factory delivery to decommissioning.

A device leaves the factory in an uncommissioned state. In this state it refuses every operational command. As part of taking possession, it accepts only the single command that installs its first credential. This must happen in a secure environment.

Above the uncommissioned state sit three base keys.

- The *Provisioning Key* is unique per device and authorizes the installation of every later credential and the destructive factory restore.
- The *Integrator Key* is shared by every node in one machine and defines the trust domain of that machine.
- The optional *Operator Key* separates day-to-day runtime authority from the integrator's deployment authority.

Operational traffic never uses these base keys directly, to limit their exposure. A server-held random salt, written once after every reset, is mixed with a base key through HKDF-SHA-256 (HMAC-based key derivation function) to produce a fresh session key for each power cycle and each of the three keys. Keys never travel on the network. Authentication tag length is dimensioned for the medium: 32 bits to 64 bits are acceptable on both CAN CC and CAN FD for selected traffic; the full 16-byte AEAD (authenticated encryption with associated data) tag is reserved for provisioning and configuration exchanges that can span multiple frames.

Figure 1 shows the salt and session key initialization after reset: Servers initialize their salt to zero to mark it as not yet set. In a write-once cycle, the client writes the salt value. From now on, this is a read-only entry, only erased by another reset. Both communication partners use that salt value to generate session keys from the three base keys. Going forward, these are now ready to be used for AEAD cryptography.

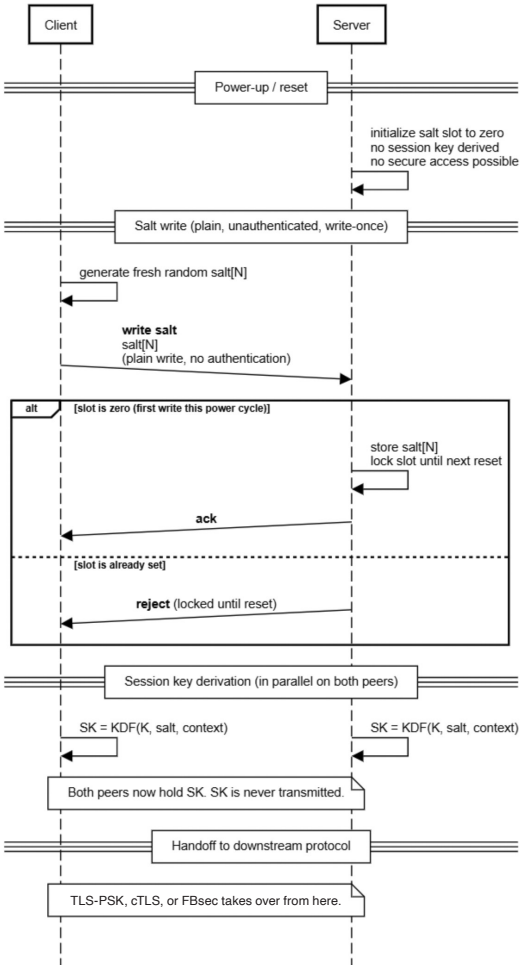

Figure 1: Deriving session keys using a shared salt value (Source: EmSA-WP-104)

SOFA: Selected secure objects over (U)SDO

SOFA is described in the white paper EmSA-WP-105. The design starts from a simple observation: on a CANopen network, the exchanges that need a cryptographic protection are mostly not the cyclic PDOs (process data objects). They are the authenticated identification, configuration, or calibration table updates and mode switches (such as Bootloader activation, etc.). These accesses are infrequent and they are already implemented as (U)SDO transfers. If we can wrap those transfers in authenticated encryption, we cover the access classes that matter for audit without rebuilding the lower stack.

From the network stack's point of view, a SOFA exchange looks like an ordinary SDO or USDO transfer against an OD (object dictionary) entry of type DOMAIN or OCTET_STRING. A communication sequence exchange is tunneled through a single object. From the application's point of view, the same entry exposes only one direction: secure-read-only or secure-write-only.

Inside SOFA there are multiple protocol options. The session-based option uses TLS-PSK (transport layer security with pre-shared key, RFC 4279) or cTLS (compact TLS) over the OD entry, with the pre-shared key being the session key derived from WP-104. After a short handshake, every subsequent access against the same object is protected by the TLS record layer. While these session-based options are best standardized, they still have significant overhead.

The per-transfer option is FBsec and requires significantly less overhead. FBsec is a stateless challenge-response exchange patterned on ISO/IEC 9798-2 Mechanism 4, truncated to two passes for unilateral authentication. Each secure read or write is a self-contained two-step transaction.

Figure 2 shows how for a secure read both peers contribute a fresh random value during the challenge step (draw_clnt_rnd, draw_srv_rnd) to protect from partial replay attempts. The nonce used for the AEAD is a combination of the two randoms and a zero initialized counter. The additional authenticated data (AAD) binds the client node-ID, the server node-ID, and the data identifier of the object being accessed. There is no session_id on the wire and no long-lived session state on the server. The default AEAD algorithm used is AES-128-GCM, as accelerators for it are available in almost every microcontroller with security enhancements.

In addition, FBsec also supports cyclic continuation without re-running the challenge, illustrated by a loop in the Figure 2. Using an incrementing counter combined with the previously generated base nonce, a session can run for up to roughly one million loops against the same object before fresh challenge material is required. No further random values need to be exchanged here. This is especially useful for continuously polling a status parameter or similar to get something like a secure node guarding.

Four concrete use cases

The point of SOFA is not the cryptography on its own; it is the set of CANopen interactions that become trustworthy once a secure object is in place. Four use cases recur in our integrator conversations.

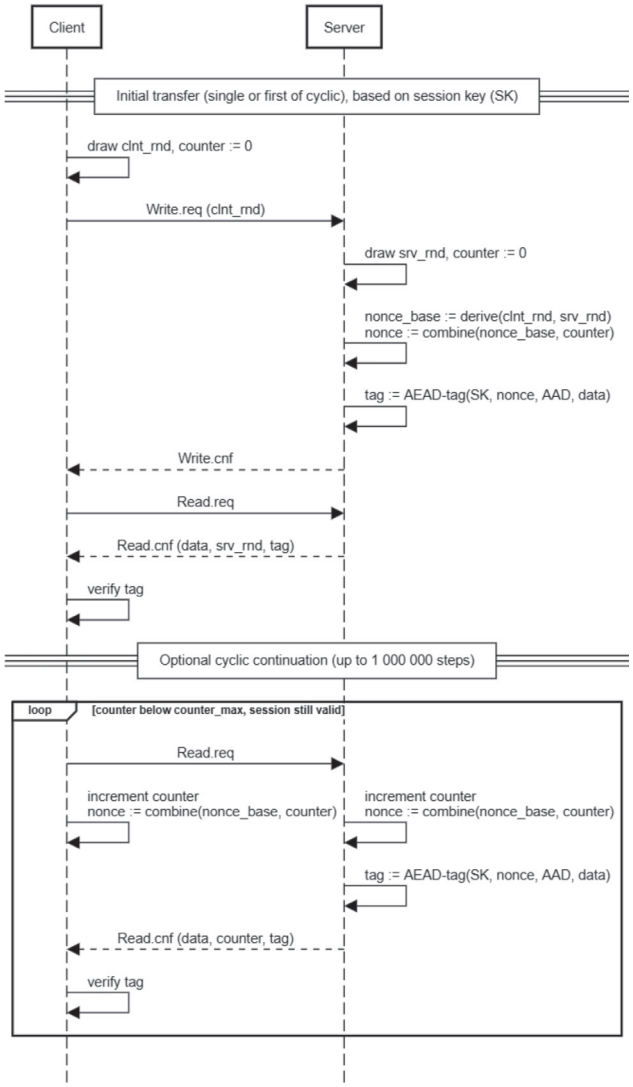

Figure 2: One FBsec secure exchange via an object dictionary entry (Source: EmSA-WP-105)

The first use case is the authenticated device identification. An OD entry exposes the CANopen vendor-ID, product code, revision number, and serial number (see object 1018h), wrapped in a SOFA secure-read. The client end of the exchange is then convinced that the answer came from the genuine device and not from a counterfeit node that has cloned the identification fields. This is exactly what supply-chain integrity language in CRA Annex I and IEC 62443 expects of a device/component.

The second use case is a secure configuration. A device exposes one DOMAIN entry that accepts the entire configuration set in a single write, typically formatted as a concise DCF (device configuration file) blob. The write is wrapped in a SOFA secure-write. Either the whole configuration arrives with a valid authentication tag and is applied atomically, or none of it is applied. There is no halfway state where half of the parameters are the new values and the rest are still the old ones.

The third use case is triggering crucial functionality, for example a mode switch into a different operating mode like a bootloader. A SOFA-protected secure-write against a defined OD entry tells the device to enter the restricted operating mode. The network stack provides the transport; the application layer holds the function behind a verified authentication tag. An unauthenticated command never reaches the mode-change logic.

The fourth use case is the configuration lock. A secure-write-only OD entry accepts one final write that transitions the device into a locked state; from that point on, further configuration writes are refused until the lock is cleared by an authenticated command under a higher-privilege key. This is the CANopen-side answer to the IEC 62443 requirement that a component must be able to refuse configuration changes outside an authenticated maintenance window.

Each of these objects shows up in the OD as an ordinary DOMAIN or OCTET_STRING entry, and each is accessed through the same SDO or USDO machinery the rest of the device already uses. Existing tools that walk the OD see nothing exotic; only the application code knows that a SOFA handler is gating the access.

Optional asymmetric identity

The reference implementation carries an optional asymmetric Ed25519 layer offering support for public/private key management systems. Following IEEE 802.1AR, a device holds a factory identity (IDevID) and generates a local one (LDevID) when it is commissioned. A manufacturer-signed ownership voucher, an idea borrowed from BRSKI (RFC 8995), names the new owner, and the device verifies it against the manufacturer key it left the factory with. Only then does it accept the Provisioning Key install that bootstraps the symmetric key tiers. The same identities can sign individual FBsec transfers, raising a secure read or write from unilateral to mutual authentication.

The SOFA reference implementation

Alongside WP-104 and WP-105, we published the matching SOFA reference implementation under a public open-source release under the Apache License 2.0. The code is written in C99 and builds with CMake. It ships with a Windows user-space simulator that allows a USDO server with a four-entry secure object dictionary and an interactive USDO client to interact with each other. An optional bridge against PEAK PCAN hardware connects the simulated mesh to a real CAN (FD) network when one is available.

All four SOFA flows are simulated end-to-end on CANopen FD: single secure read, single secure write, and the cyclic-polled variants of both.

The two white papers, WP-104 and WP-105, and the reference implementation is published at: https://github.com/em-sa/SOFA-CANopenFD.

Author

Olaf Pfeiffer
Emsa (Embedded Systems Academy)
info@esacademy.com
www.esacademy.de