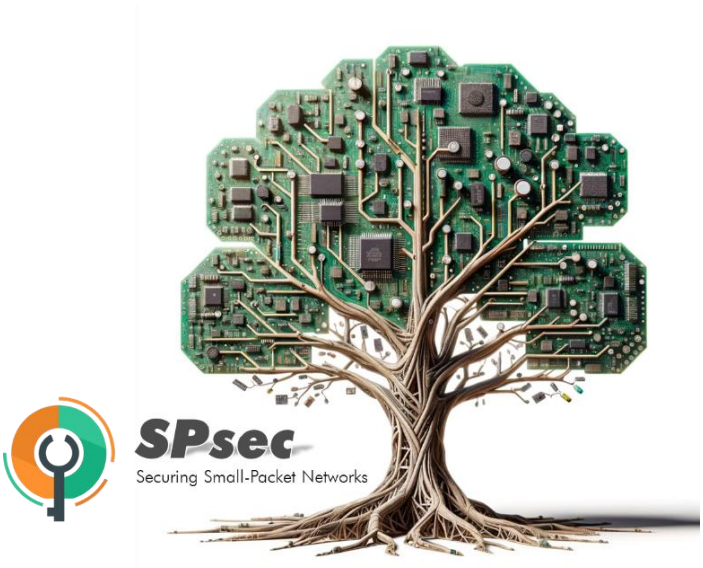
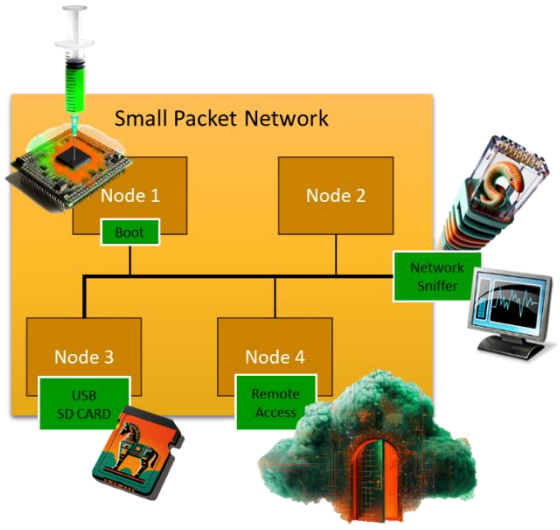


Cybersecurity Primitives for Small-Packet Networks

This work is based on a research grant, collaborative project of the Institute of Reliable Embedded Systems and Communication Electronics (ivESK, Prof. Sikora of Offenburg University) and the Embedded Systems Academy. It has been awarded focusing on embedded network security. The project is dedicated to developing a security framework for small-packet networks, with a specific emphasis on CAN and CANopen systems. This project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag.



SMALL-PACKET NETWORK ATTACK VECTORS

Regulatory Demands

Around the world, tougher regulations and acts are being discussed and enacted, which in the strictest case stipulate that electronics-based systems must follow “security by design”. Some specifically define that(relevant) data in rest and in transit needs to be protected by state-of-the-art authentication and encryption. Consequently, if data exchanged via small-packet networks like LIN, I2C, Modbus, CAN, CANopen or other fieldbuses must meet the requirements of such regulations, it needs to be protected in several applications.

Cryptographic Primitives

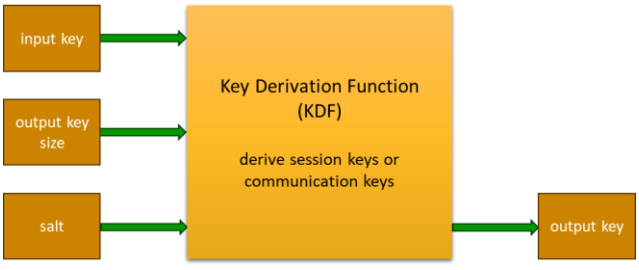
One of the challenges in securing small-packet network communication is the lack of bandwidth and the prevalent limited CPU resources. Nevertheless, some basic cybersecurity requirements must be met. In summary, the following key points are addressed:

- Minimal hardware requirements
- Cryptographic functions
- Point-to-point security
- Time-based rolling key derivation
- Group security

Minimal Hardware Requirements

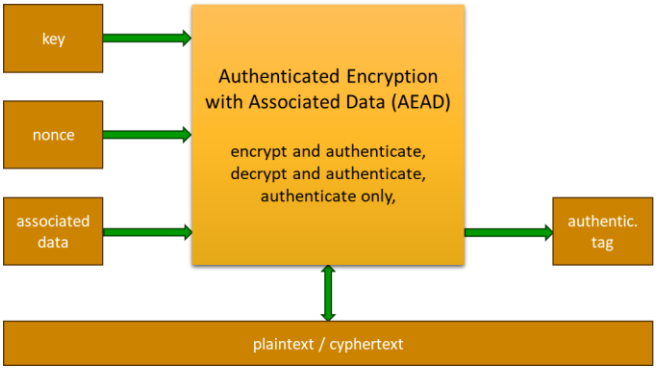
The usage of cryptographic methods requires more resources from the used microcontroller-based systems. Besides the processing power needed for the selected cryptographic algorithms, further requirements are a true random number generator and secure storage for the used pre-shared cryptographic key(s). If a security hardened microcontroller is not available, it must be carefully determined if and how the requirements for true randomness, secure key storage and processing power can be adequately met.

Cryptographic Methods and Functions: KDF, AEAD



KDF INPUTS AND OUTPUTS

For key derivation standardized KDF (key derivation functions) are used. Authentication and encryption are AEAD based (authenticated encryption with

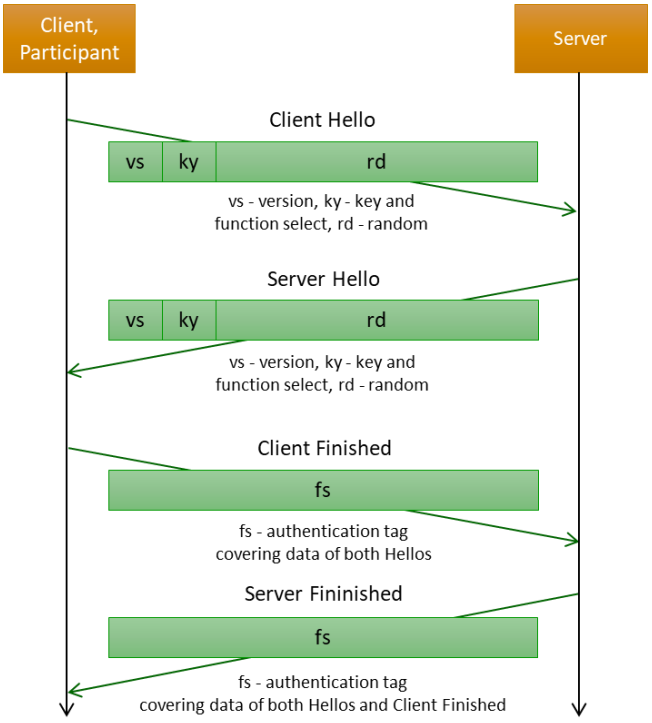


AEAD INPUTS AND OUTPUTS

associated data), providing a standardized method to produce authentication tags and optionally encrypt data. Selection of algorithms and tag sizes are to be specified in accordance with the resources available to the participants in the communication and the additional bandwidth made available for the security overhead (adding authentication tags to communication objects).

Point-to-Point Security with microTLS

With microTLS, a variation of TLS-PSK (Transport Layer Security with pre-shared keys), point-to-point security can be established. This method allows establishing a secure communication channel within a total of 4 messages (each as small as 8 bytes for classical CAN or 8-10 bytes for CAN FD) exchanged.

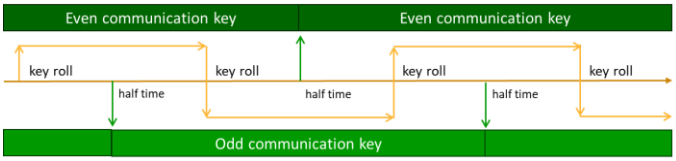


MICROTLS DERIVED FROM TLS-PSK

Based on the secure channel, both Client and Server agree on a shared session key and an initial counter value usable as uniqueness/nonce input to the AEAD function. Each communication exchange includes the least significant portion of the counter (to ensure both Client and Server are still in sync and to prohibit replay attacks) and an authentication tag.

Time-based Rolling Key Derivation

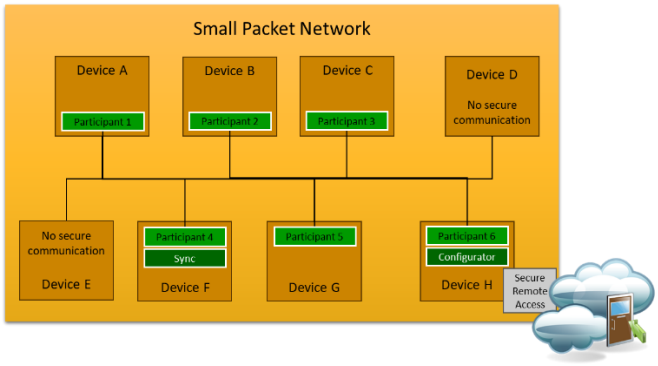
To minimize key exposure, the current communication keys are automatically derived from the main pre-shared key on a fixed time basis. This requires a synchronized time basis among all the participants in the secure communication. To avoid the need to do an exact synchronized key change, there are always two keys valid: the current and either the previous or the next.



OVERLAPPING ROLLING COMMUNICATION KEYS

If a new key is derived every hour, then to allow for time deviations among the participants, it shall be valid 30min before and 30min after its main window of validity (half time). This method allows participants to not worry about the exact time the next key becomes valid, and they can already generate the next key well before it will be used for the first time.

Group Security



In group security, all Participant roles in secure communication use the same time-based derived rolling communication keys. Therefore, any member of the group can consume and produce any part of secure communication. The time synchronization is performed by a single Sync role. The optional

Configurator role is only used during setup and maintenance to configure the system.

Transmitters use a combination of the current timestamp and a network/message address as uniqueness/nonce input to the AEAD function. Each communication exchange includes the least significant portion of the timer (will be restored to full length and can used to determine if arrival time is within an acceptance window) and an authentication tag.

Applying Small-Packet Network Security to CAN FD / CANopen FD

For mapping the above-mentioned methods to CAN FD and CANopen FD, the goal is to have all devices connected being participants in encrypted and authenticated communication. Therefore, all devices require a pre-shared key.

For the KDF and AEAD function there are currently various cryptographic methods reviewed for their performance and resource requirements. The candidates include AES-GCM, ChaCha20-Poly1305, Ascon for AEAD and HKDF/SHA-256 for KDF.

The last 8 bytes of any CAN FD / CANopen FD frame are used for holding a security stamp holding a combination of the current portion of the synchronized timer and a 48-bit authentication tag. Note that in CANopen FD there is a maximum payload size of 64 byte. Since the security stamp occupies 8 bytes, the actual payload length needs to be reduced to 56 bytes. With that setting, CANopen FD will not produce CAN FD frames with a payload larger than 56 bytes.

The microTLS implementation uses 64-bit values for randomness from both Client and Server and 64-bit authentication tags in both finished and data transfers. For classical CAN these could be reduced to 48bit.

The synchronized time is a randomly initialized by the sync role responsible for maintaining the, synchronized, free-running 64-bit timer incrementing every 100 microseconds. Upon initialization participants use a microTLS transfer to authenticate the initial synchronized timestamp received.

Communication key derivation happens on overrun of bit 24 of the 64-bit timer, so roughly every 30 minutes. The upper bits 24 to 63 are used as salt input to the KDF when generating the rolling communication key matching a particular timestamp.

About this Document

By
Olaf Pfeiffer of Embedded Systems Academy GmbH
August 2024.

Embedded Systems Academy
www.em-sa.com, info@esacademy.de

Institute of Reliable Embedded Systems
and Communication Electronics
(ivESK) Prof. Axel Sikora

This Project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag.