SPsec201 – Generic Specification

This document defines the basic data types and services of SPsec (Small-Packet Network Security Sublayer).

The SPsec specifications are divided into the following documents:

- **SPsec101** – Small-Packet Network Security Concept
  Introduction to the security concept of SPsec.
- **SPsec102** – Small-Packet Network Security Glossary
  Terminology and references used by SPsec.
- **SPsec201** – Small-Packet Network Security Generic Specification
  Network independent specification of the SPsec technology and methods.
- **SPsec301** – Small-Packet Network Security Serial Channel Mapping
  Mapping SPsec to generic serial point to point communication.
- **SPsec302** – Small-Packet Network Security CAN FD Mapping
  Mapping SPsec to CAN FD, supporting CANopen FD and J1939 FD.

Version 1.35 of 11-DECEMBER-2025, jointly authored by

www.em-sa.com
Embedded Systems Academy GmbH
Bahnhofstraße 17
30890 Barsinghausen, Germany

ivesk.hs-offenburg.de
Hochschule Offenburg
Badstraße 24
77652 Offenburg, Germany

This project has been funded as part of the Central Innovation Program for SMEs (ZIM) by the Federal Ministry for Economic Affairs and Climate Action (BMWK).

Contents

1 General Settings

In the mapping document this section defines the overall network specific settings. These may include parameters like bitrates, addressing scheme, maximum sizes of data packets exchanged and methods for assigning the Participant IDs.

2 Constants, Data Types and Parameters

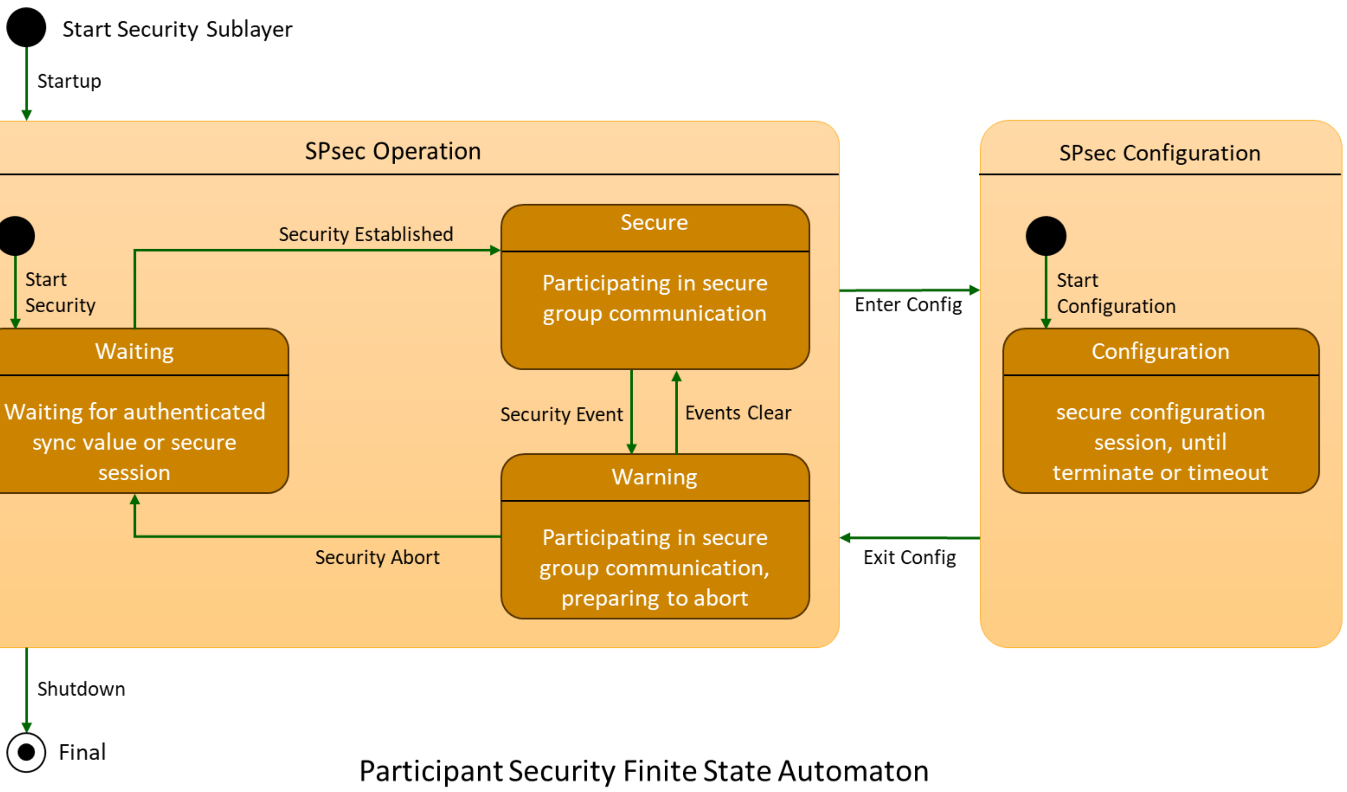
This section defines the core parameters used by SPsec. The corresponding network mapping document defines data type details specific to a network technology.

2.1 Participant ID

This unsigned integer value is used to uniquely address or identify a Participant. If the network technology used has node IDs, then the Participant ID is the same as the node ID.

The network mapping document defines the specific size and value range.

2.2 SPsec Participant State



Participant Security Finite State Automaton

Every Participant implements a finite state automaton (FSA).

Note that the communication interface initialization must happen before starting this state machine.

2.2.1 SPsec Participant State Groups

SPsec Operation: In these states the Participant actively tries to reach and maintain the Secure state. In any of these states the Participant accepts requests from the Configurator.

SPsec Configuration: The Participant is currently being configured by the Configurator until it receives a terminate request or until the configuration session times out.

2.2.2 SPsec Participant States

Waiting: In this state, a Participant with available Seed key waits until it has received the authenticated sync value / timer. If no Seed key is available, it waits for a configuration session.

Secure: A Participant in this state actively participates in the secure group communication.

Warning: A Participants in this state detected a security warning, like a secure heartbeat timeout or an authentication failure. Depending on configuration and severity of the warning, the Participant may resume or abort.

Configuration: A Participant in this state is currently configured by the configurator through an active SPsec session. No other communication is happening in this state.

2.2.3 SPsec Participant State Transitions

2.2.3.1 State Group Transitions

Startup: Start from power down or reset.

Enter Config: Start of a SPsec Session based configuration session.

Exit Config: Termination or timeout of the configuration session.

Shutdown: Shutting down the security FSA, should only be done when communication is switched off or system powers down.

2.2.3.2 Internal State Transitions

Start Security: Starting the SPsec functionality. Successor state to “Startup” or “Exit Config”.

Start Configuration: A configuration session with the Configurator was established. Successor state to “Enter Config”.

Security Established: The initial Sync / Time value was received AND authenticated.

Security Event: The Participant detected a single security warning like a timeout or a mismatch of a cryptographic checksum.

Events Clear: The warning reason was a single occurrence, and no further warning reason occurred within the warning state hold time.

Security Abort: A sync time synchronization timeout or a sync time synchronization key failure occurred.

2.3 Parameter Register Number

This unsigned integer value (8 bits, unless specified differently in mapping document) is used to uniquely address a parameter (register) within a Participant. The access type is conditional. As an example, keys may never be read, but might be written once during initial configuration, if not yet set.

Value ranges pre-select the type of information stored in a register and how they can be accessed by the Configurator.

00h-1Fh:	Reserved
20h-2Fh:	Keys (WO)
30h-3Fh:	Key Salt (WO)
40h-4Fh:	Key ID (RW)
50h-5Fh:	SPsec Information (RO)
60h-7Fh:	SPsec Configuration (RW)
80h-8Fh:	Device Information (RO)
90h-9Fh:	Code Updates (RW)
A0h-CFh:	Reserved
D0h-EFh:	Manufacturer Specific (RW)
F0h-FFh:	Reserved

The mapping document defines additional details like data type, size of each parameter and additional, network specific parameters.

2.3.1 Registers 20h to 2Fh: Configurator Session Keys and Seed Key

These registers can never be read. See section 2.4 for details on write access requirements.

2.3.1.1 21h: Provisioning Key

This is a conditional write-only register, see section 2.4 for details.

2.3.1.2 22h: Integrator Key (WO, conditional)

This is a conditional write-only register, see section 2.4 for details.

2.3.1.3 23h: Seed Key (WO, conditional)

This is a conditional write-only register, see section 2.4 for details.

2.3.2 Registers 30h to 3Fh: Key specific pre-shared salt

Some cryptographic functions require salt or a nonce of a specific length. In those cases where the values available (from counters or timestamps) are not long enough, pre-shared salt shall be added. See section 2.4 for details on write access requirements. Details are defined in the network mapping document.

2.3.2.1 31h: Provisioning Key salt

This is a conditional write-only register, see section 2.4 for details.

2.3.2.2 32h: Integrator Key salt

This is a conditional write-only register, see section 2.4 for details.

2.3.2.3 33h: Seed Key salt

This is a conditional write-only register, see section 2.4 for details.

2.3.3 Registers 40h to 4Fh: Key ID

A unique key ID is associated with every key. This information can be used by configuration and diagnostic tools to share some ID information about keys. These registers can always be read. See section 2.4 for details on write access requirements.

2.3.3.1 41h: Provisioning Key ID

This is a conditional write-only register, see section 2.4 for details.

2.3.3.2 42h: Integrator Key ID

This is a conditional write-only register, see section 2.4 for details.

2.3.3.3 43h: Seed Key ID

This is a conditional write-only register, see section 2.4 for details.

2.3.4 Registers 50h-5Fh: SPsec Information

These are read-only registers providing information about the SPsec implementation and status.

2.3.4.1 50h: SPsec Status

The data type for the SPsec Status is an unsigned integer of 8 bits or an enumeration with a combination of the current state (FSA) of a Participant in the lower 4 bits and further information in the upper 4 bits.

Bits 0-3: Current state

0: Not set
1: Waiting
2: Secure
3: Warning
4: Configuration
5: Shutdown
other: reserved

Bit 4-6: Reserved

Bit 7: Alert - set, by transition “Security Event”. Reset on entering state “Waiting”

Changes to this register can trigger an event on the internal or external control plane sharing the content of this register. Note that when read from a Configurator, content is always 4. Further details are specified in the mapping document.

2.3.4.2 51h: SPsec Last Security Event

Updated with each security event detected. Changes to this register can trigger an event on the internal or external control plane. Details are specified in the mapping document.

2.3.4.3 58h: SPsec Core Version Information

Returns a string with the version number of SPsec201 specification implemented. For example “1.29”.

2.3.4.4 59h: SPsec Mapping Version Information

Returns a string with the mapping document number and version number of SPsec30x specification implemented, separated by a hyphen. For example "302-1.29".

2.3.5 Registers 60h-7Fh: SPsec Configuration

These registers hold the current SPsec configuration. If Participant behaviour, timeouts, usage of addresses are configurable, the mapping document defines them in this register area.

2.3.5.1 60h: Participant ID

The Participant ID of this device. A new value written is activated upon the next "Start Security" transition and is stored in non-volatile memory.

2.3.5.2 61h: Secure Heartbeat Timing

This parameter determines the cycle time used to produce the secure heartbeat. A new value written is activated upon the next "Start Security" transition and is stored in non-volatile memory.

All participants in a SPsec network must use the same Secure Heartbeat Timing setting. Depending on application this may also be hard -coded and implemented as read-only register.

The mapping documents defines they heartbeat cycle time and the timeout used.

2.3.5.3 62h: Secure Heartbeat Monitoring

This optional parameter determines which heartbeats are monitored by a Participant. If not defined, all heartbeats shall be consumed.

The mapping documents defines they heartbeat cycle time and the timeout used.

2.3.5.4 63h: Sync Role Activation

Implemented, if the device has the capability to execute the sync role.

2.3.5.5 7Fh: Manufacturer Reset to Default

This register is used to reset a device into the Manufacturer Default state. All configuration settings and keys besides the Provisioning Key are erased.

2.3.6 Registers 80h-8Fh: SPsec Device Information

These registers are read-only. During a SPsec Session connection based on any key, a configurator can read these registers. This information is provided by or stored in the SPsec sublayer. Should be set upon initial configuration of the device. The network mapping document defines the specific size and additional values.

2.3.6.1 81h: Device Identification

Returns the device's identification as string. The string shall include information about the manufacturer, the product (code) and the serial number. Further details are defined in the mapping document.

If this information is not directly available to the SPsec sublayer, then this register is implemented as a write once register. During initial setup, a Configurator can write the device identification string to this register.

2.3.6.2 82h: MCU Serial Number

Returns the microcontrollers's serial number as string.

2.3.7 Registers 90h-9Fh: Code Updates

These optional registers provide information and access to securely load code updates. The configurator shall only write a code update, after verifying the authenticity of the update file using the public authentication key. Upon power cycle, the bootloader firmware verifies, if the buffer of the Code Update register holds a valid code update. Details about what makes a code update file valid are manufacturer specific.

2.3.7.1 90h: Code Update Capabilities

This read-only register provides information about the code update capabilities of this device.

2.3.7.2 91h: Public Authentication Key

This read-only register holds the public key that can be used to authenticate code update files.

2.3.7.3 92h: Code Update File

The code update file needs to be written to this write-only register. Data written to this register is stored in a sperate memory area and only processed with the next power cycle.

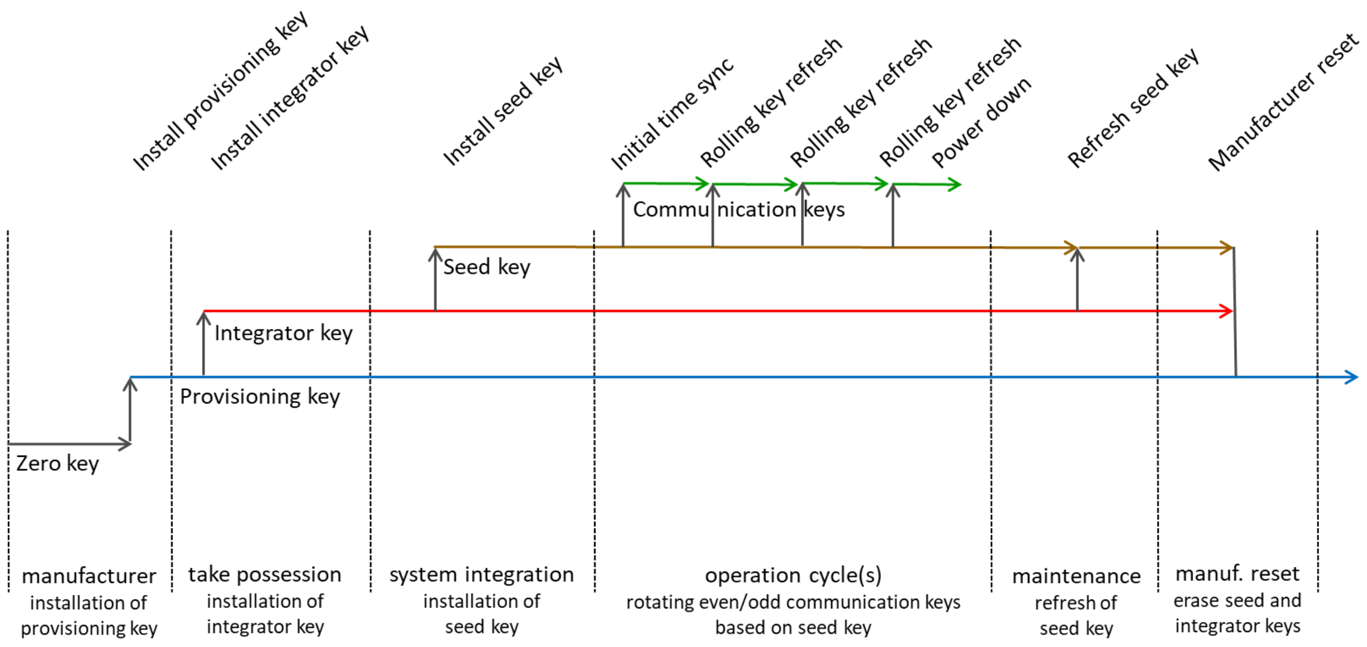
2.4 Keys

The following key sets (consisting of Key, Salt and Key ID) and keys are supported by SPsec, sorted by highest trust first:

- Provisioning Key, Salt and Key ID:
  The Provisioning key is a device-specific root style key used for Configurator sessions to read or write registers. This key set can only be written once. The initial write must either happen manufacturer specific (outside of SPsec) or in a trusted environment using a Configurator session based on the Zero key.
  The Key ID can be read by a Configuration session based on any key (including Zero key).
- Integrator Key, Salt and Key ID:
  The Integrator Key is the Integrator's root style key. It is used for Configurator sessions to read or write registers. This key set can only be written once. The initial write must happen using a Configurator session based on the Provisioning key.
  The Key ID can be read by a Configuration session based on any key (including Zero key).
- Seed Key, Salt and Key ID:
  The Seed Key is the base key for Communication Key derivations. The key set can be written and over-written using a Configurator session based on the Integrator key.
  The Key ID can be read by a Configuration session based on any key (including Zero key).

- Parameter Authentication key:
  A temporary session key to authenticate a single parameter such as a counter or timestamp. Auto-generated and discarded after one-time use. Not available through registers.
- Session Key:
  A temporary key used during the Configurator session. Auto-generated and discarded after session termination. Not available through registers.
- Communication Keys:
  Temporary keys used to protect the addressed data units. Auto-generated (refreshed) on a time or counter basis from the Seed key. Not available through registers.
- Zero Key:
  This key consists of all zeros and is used for unsecure Configurator sessions to read any key ID provided in registers. The unsecure Configuration session based on the Zero key is used to write the initial Provisioning key set.

2.5 Key Installation



- The Provisioning key is installed by the manufacturer of a device and passed on to the integrator, for example printed on the serial number label of the device.
- The integrator takes possession and sets the Integrator key. Depending on use case this can be a device specific key or a pre-shared group/network key.
- Upon integration of the device, the integrator sets the pre-shared seed key. This must be shared among all participants in one network.
- During operation, odd and even communication keys are automatically derived from the seed key.
- The seed key should be updated with every maintenance session

- In a secure session based on the Integrator key, a manufacturer reset can be triggered. This erases both the seed and integrator key. In this stage, a new integrator key can be installed based on a session using the provisioning key.

Upon system integration, a secure session based on the Provisioning key is established to set the Integrator key.

2.6 Pre-shared Salt

Some cryptographic functions might require a pre-shared salt. Where supported, this is generated and assigned together with the key assignment.

2.7 Key ID

The Key ID is a 32bit unsigned integer value associated with a key. The value may not be all bits cleared or all bits set.

If a configurator writes a key to the Participant, it shall also generate a unique Key ID for that key and store it in the Participant. Key and Key ID are a pair, each unique key value shall have its own unique key ID. Keys cannot be read out, but Key IDs can. If a configurator has access to a database of keys assigned along with their Key IDs, then it can use the Key ID to request the matching key from the database.

To ensure integrity, the sequence for a configurator to write a key is as follows:

1.) Erase key ID (set to FFFFFFFFh)
2.) Write the Key
3.) Write the associated Key ID

Key IDs 0 and FFFFFFFFh are reserved and indicate that there is no valid key available.

2.8 Key Selector

This unsigned integer value is used to select one of the keys defined by SPsec.

0: Reserved
1: Zero key
2: Session key (for 1:1 sessions)
3: Even Communication Key
4: Odd Communication Key
5: Parameter Authentication Key
6-12: Reserved
13: Seed Key
14: Integrator Key
15: Provisioning Key

Key selector 2-5 indicates temporary keys not stored in non-volatile memory.

There are two communication keys defined to allow two keys to be active during a transition time from one communication key to the next. For example, if communication keys are generated on an hourly basis, the even key is used for even hours, the odd for odd hours.

The network mapping document may define optionally additional values.

2.9 Security Event and Codes

This unsigned integer value is used by Participants to indicate a security event. There are three value ranges:

00h: Events Clear, positive event, previous warning resolved or reset
10h-E0h: Security event
F0h: Security abort

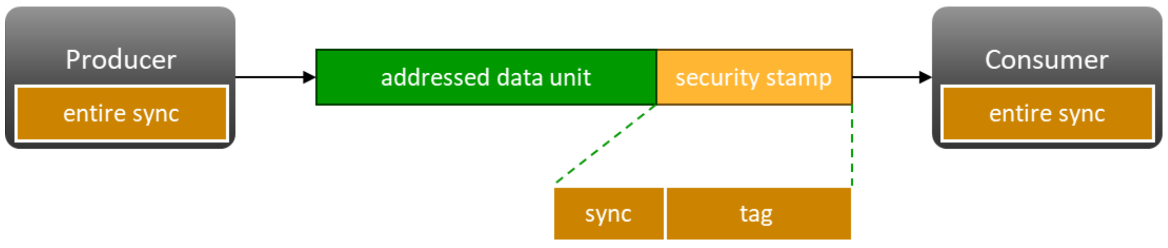
These value ranges directly correspond to the three state transitions in and out of the Warning state of the Participant Security FSA.

The network mapping document defines the specific size and additional values.

2.10 Security Stamp

The Security Stamp is a data structure containing the information added to each addressed data unit to make it a secured addressed data unit. It consists of an unsigned integer uniqueness value and an unsigned integer authentication tag (cryptographic checksum).

The network mapping document defines the specific structure and sizes.



sync: portion of entire sync value (typically least significant bits)
tag: authentication tag covering the entire sync value and the entire addressed data unit (meta and data)

If the uniqueness value is synchronized automatically (like an increasing counter), then the Security Stamp does not contain the entire uniqueness value but may be truncated to the lowest bits.

2.11 Uniqueness Value: Shared Message Counter

Counters are used as uniqueness value for single channel, point to point communication, based on a client / server model. Unless specified differently in the network mapping document, this is an unsigned 32bit value which shall be initialized securely with a random value during the initial client / server communication to ensure an unpredictable start value. This can be done using a SPsec session. In further exchanges, both client and server increment the counter before each transmission of a secured addressed data unit.

2.12 Uniqueness Value: Global Synchronized Timestamp

A synchronized timestamp is used as uniqueness value for grouped communication where the same addressed data unit is a broadcast or multicast to multiple Participants. By default, this is a free running timer.

The network mapping document defines the sizes and resolution used. The resolution should be such, that every secure addressed data unit exchanged has its own timestamp. The resolution should not be higher than that, as a higher resolution contributes to a higher number of bits being exchanged with every Security Stamp.

2.12.1 Enforcing Uniqueness

When using a free-running timer as a base for the timestamp, some extra effort needs to be taken to ensure that timestamps do not repeat, as that can break AEAD based security. For example, re-using a nonce for AES is a known vulnerability.

The responsibility for enforcing the uniqueness lays within the Timer Sync Role.

If the timer is initialized randomly, then the accidental chance of a re-use depends on the resolution and total size of the timestamp. A randomly initialized timer should not be used unless its size is at least 128bits or more.

A more secure implementation ensures that the timestamp reflects the total runtime. It is reset with setting the seed key and continuously incremented, even across power cycles. This can be implemented using secure NVOL storage where the timestamp used is stored cyclically. On every power-up, the last value is retrieved and incremented by a value greater than the storage cycle time, ensuring that no value can be re-sued until timer overrun.

2.13 Control plane Message Types

This value indicates the different message types supported by the control plane. In the mapping document these can be mapped into the address part or into the data part of the addressed data units.

SPsec Configurator SPsec session:

CPMT_SESS_HELLO: Session "Hello"
CPMT_SESS_FINISH: Session "Finish"
CPMT_SESS_RDINIT: Initiating a register read
CPMT_SESS_RDSEG: Register read data segment
CPMT_SESS_WRINIT: Initiating a register write
CPMT_SESS_WRSEG: Register write data segment
CPMT_SESS_TERMINATE: Terminating the session

SPsec parameter authentication and synchronization:

CPMT_AUTH: Parameter authentication
CPMT_SYNC: Broadcast synchronization

SPsec device or system status:

CPMT_HB: Device Secure Heartbeat
CPMT_INTERN_EVT: Device Internal Security Event

3 Cryptographic Primitives

In this section we list the cryptographic primitives required by SPsec.

Preferably these are encapsulated in a separate hardware security chip or module to minimize key and parameter exposure.

If a hardware security module cannot be used, use a security hardened and optimized microcontroller that allows limiting key exposure. If it supports a “trusted zone”, place the permanent keys (Provisioning, Integrator & Seed) and the cryptographic functions into the trusted zone. Instead of keys (or pointers to them) the cryptographic functions should use the key selector to identify the key used.

3.1 True Random Number Generator

Cryptographic functions require a true random number. Purely software based algorithms without hardware provided entropy are not suitable to achieve higher security levels.

In some cases, software based random numbers might be usable, if the seed used is non-reproducible. Sometimes this can be achieved from a combination of analogue inputs, timer variations and NVOL memory contents including previously used seeds.

3.2 Secure Key Storage

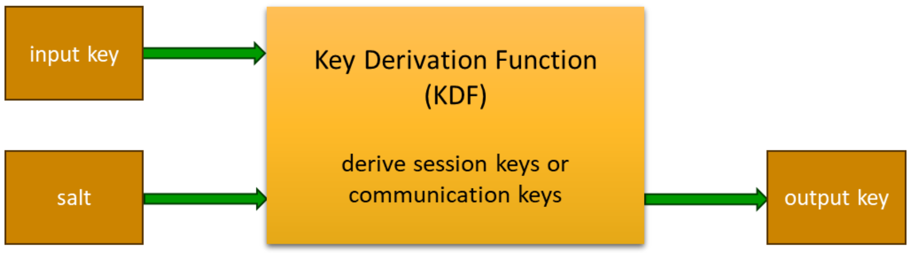
The microcontroller and software implementing the SPsec functionality must limit the exposure of the keys used. All keys must be stored in a way, that they can not easily be read out, even if an attacker has unlimited physical access to the storage memory.

In cases where secure memory storage is limited, that shall hold a device specific key which is used to encrypt and authenticate all SPsec keys stored in NVOL memory without specific protection.

3.3 KDF – Key Derivation Function

The KDF is used to generate temporal keys.

- A SPsec Session key gets derived from the Integrator or Seed key and random or nonce data provided as salt by both communication partners.
- In grouped security, the current communication key gets derived from the Seed key and the timestamp used as salt. By default, a new communication key is derived every hour, on the hour.


input key
salt
Key Derivation Function (KDF)
derive session keys or communication keys
output key

Inputs: Base key and salt value as well as the size of the output key.

Outputs: The key derived.

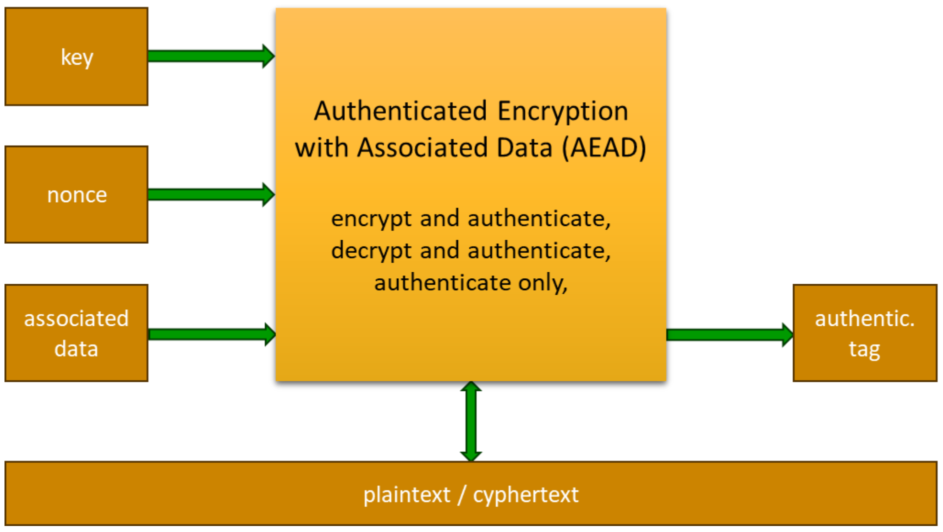
If pre-shared salt is associated with the keys, then the output key inherits the pre-shared salt from the input key.

The network mapping document defines the specific data sizes and cryptographic functions used by the KDF.

3.4 AEAD – Authenticated Encryption with Associated Data

An AEAD scheme is used to protect the addressed data units.

- In a SPsec Session, it uses the session key, the sequence counter as nonce and optional address and size information as associated data.
- In grouped security , it uses the communication key, the timestamp as nonce and optional address and size information as associated data.


key
nonce
associated data
Authenticated Encryption with Associated Data (AEAD)
encrypt and authenticate,
decrypt and authenticate,
authenticate only,
authentic. tag
plaintext / cyphertext

Inputs: Encryption and authentication key (session key or communication key), counter or timestamp, address meta data as associated data and the data itself.

Outputs: The encryption/decryption of the data and the authentication tag.

NOTE: If encryption/decryption is not used, then data input/output is NOT used. In this case pass add the data to the "associated data" to get an authentication tag calculated.

The network mapping document defines the specific inputs, data sizes and cryptographic functions used by the AEAD.

4 Key derivations

To minimize key exposure, where possible, keys are derived from pre-shared keys (PSK).

4.1 Zero Key

An unsecure initialization key used to read key IDs or to write the initial provisioning key.

4.2 Provisioning Key

As this is a device unique key, it is never used for any key derivations.

4.3 Integrator Key

Integrator key usage is application specific. It can be unique to a device or shared with multiple devices depending on the use case. It is used to derive session keys used by secure configuration sessions.

4.3.1 Used as Device Unique Key

If used device uniquely, integrator's can ensure that every device requires its own secure connection for future configurations.

4.3.2 Used as Group Key

If used shared among all devices in one network, diagnostics and maintenance can be simplified as only one key is needed to access and configure all nodes of one network.

4.4 Seed Key

The seed key must be pre-shared among all SPsec participants in the same network. It is the base for Communication key derivations.

4.5 Odd and Even Communication Key

These keys are automatically derived on a fixed time basis. The input key is the Seed key and the salt is a truncated timer value combined with a Communication Key Derivation Salt. Details are specified in the mapping document.

4.6 Session Key

Session keys are used for time authentication and SPsec Sessions. The input key is either the Zero, Provisioning, Integrator or Seed key and the salt is a combination of two "Hello" values exchanged in the SPsec Session. Details are specified in the mapping document.

4.7 Parameter Authentication Key

The parameter authentication service uses a one-time, one-use temporary key derived from one of the other keys.

5 Cryptographic Functions and Modules

This chapter summarized the cryptographic functionality provided by SPsec. The related SPsec protocols are defined in the next chapter.

5.1 SPsec Session

The SPsec Session is a secure client-server session between two Participants based on one of the pre-shared keys.

The primary use of this communication method is to provide a secure method for configuration, where the Configurator role is the client and the Participant being configured is the server.

The session is based on concepts from TLS-PSK (Transport Layer Security with Pre-Shared Keys) and cTLS (Compact TLS) , but optimized for the SPsec use cases. The exchanges between client and server are:

- Hello: Derive a session key
- Finished: Verify session key
- R/W: Read and write configuration access
- Terminate: Terminate the session

There is an overall session timeout and a response timeout to ensure that sessions are terminated, if the communication partner gets disconnected.

5.2 SPsec Parameter Authentication

The SPsec Parameter Authentication is used to authenticate a single parameter.

The primary use of this communication method is to authenticate the synchronized timer value. The Participant requesting the timer value acts as client and the Timer Sync Role acts as server.

Once the initial value has been synchronized, further synchronization are based on data plane style protection using secured addressed data units.

This protocol is a further optimization of the methods used in the SPsec Session and consists of a single request and response:

- Client Hello: Requesting time value
- Server Hello & Finished: Contains Hello, Finished and time value

5.3 SPsec Multi-Participant Grouping

The SPsec Multi-Participant Grouping is the main secure communication mode provided to exchange the process data (data plane communication) using secured addressed data units.

This communication mode requires, that all participants have a synchronized uniqueness value (counter or timer, depending on implementation) and a current Communication key. The Communication key is automatically derived based on the synchronized uniqueness value (whenever counter / timer overruns a pre-defined value). This ensures that all participants know when and how to refresh the Communication key.

5.4 SPsec Heartbeat

In network systems supporting multicast or broadcast communication, the SPsec Heartbeat is used by all Participants to cyclically share their current security state.

The service is based on secured addressed data units exchanged during SPsec Multi-Participant Grouping.

The specific heartbeat times are defined in the network mapping document.

In systems with limited resources (both computational and bandwidth), this may be the only secure communication possible. In this case all Participants must use non-cryptographic methods to check network integrity (detect injections or other manipulations) and stop their heartbeat upon such a detection. Therefore, any heartbeat loss has to be seen as a potential attack. Addressed data units received from a device without a heartbeat present must be ignored or trigger a security event.

6 External Control Plane Services

The services of the external control plane provide dedicated SPsec communication channels independent from the regular data plane communication on the destination network. Depending on the mapping document, this can be ensured by using reserved addresses in the addressed data units allowing a distinction between control plane or data plane content.

Any control plane message received by a participant is evaluated by verifying:

- Message type
- Parameters for that message type (counter, key select, other)
- Expected sequence (response to a request)
- Verification of authentication tag (where used)

Any detected mismatch or failure is reported to the internal control plane and optionally logged as a security event. The event is NOT reported to the external control plane as to not provide a potential attacker with feedback.

If security events are detected during a protocol sequence (of requests and responses), the detecting party aborts the sequence. To the communication partner, the absence of the next message in the sequence will trigger a timeout.

6.1 SPsec Session Start

This service starts a secure client/server session between two Participants. The default use case described here is that of a Configurator and a single Participant. The network mapping documents may introduce additional use cases.

The secure session key and the initial counter value (for uniqueness) are derived from the selected base key (by Key Selector) and the random inputs of both Configurator and Participant.

The session key derivation uses the methods defined by the SPsec Functionality selector. Inputs are the selected key and the entire data field from client and server Hellos.

Parameters of ClientHello():

- mt: Message Type (CPMT_SESS_HELLO)
- pid: Participant Node ID
- ks: Key Selector (Seed key, Integrator key, Provisioning key, Zero key)
- rnd: Random

Parameters of ServerHello:

- mt: Message Type (CPMT_SESS_HELLO)
- pid: Participant Node ID
- rnd: Random IV

Parameters of ClientFinished():

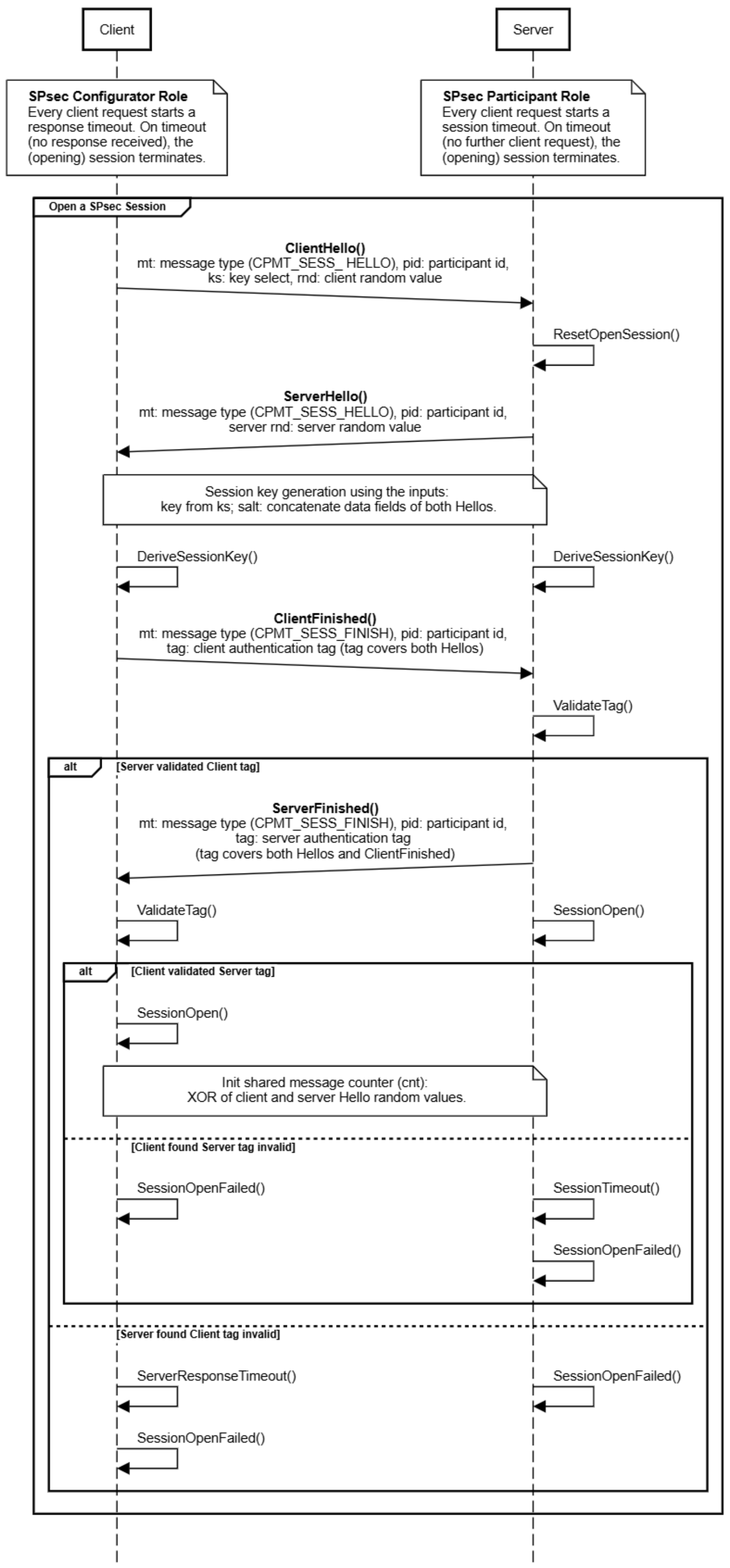
- mt: Message Type (CPMT_SESS_FINISH)
- pid: Participant Node ID
- tag: Authentication tag (covers data fields from both Hellos, details in mapping document)

Parameters of ServerFinished():

- mt: Message Type (CPMT_SESS_FINISH)
- pid: Participant Node ID
- tag: Authentication tag (covers data fields from both Hellos and ClientFinished, details in mapping document)

Opening a SPsec Session

Generic version not mapped to a specific communication technology


Client
Server
SPsec Configurator Role Every client request starts a response timeout. On timeout (no response received), the (opening) session terminates.
SPsec Participant Role Every client request starts a session timeout. On timeout (no further client request), the (opening) session terminates.
Open a SPsec Session
ClientHello() mt: message type (CPMT_SESS_ HELLO), pid: participant id, ks: key select, rnd: client random value
ResetOpenSession()
ServerHello() mt: message type (CPMT_SESS_HELLO), pid: participant id, server rnd: server random value
Session key generation using the inputs: key from ks; salt: concatenate data fields of both Hellos.
DeriveSessionKey()
DeriveSessionKey()
ClientFinished() mt: message type (CPMT_SESS_FINISH), pid: participant id, tag: client authentication tag (tag covers both Hellos)
ValidateTag()
alt
[Server validated Client tag]
ServerFinished() mt: message type (CPMT_SESS_FINISH), pid: participant id, tag: server authentication tag (tag covers both Hellos and ClientFinished)
ValidateTag()
SessionOpen()
alt
[Client validated Server tag]
SessionOpen()
Init shared message counter (cnt): XOR of client and server Hello random values.
[Client found Server tag invalid]
SessionOpenFailed()
SessionTimeout()
SessionOpenFailed()
[Server found Client tag invalid]
ServerResponseTimeout()
SessionOpenFailed()
SessionOpenFailed()

The session shared message counter initialization is a truncated XOR of both IVs from client and server.

The network mapping document defines how the parameters are mapped into addressed data units as well as the following timeouts:

SPsec Session response timeout – the maximum time allowed between any of the “Hello” and “Finished” exchanges (typically tens of milliseconds). This timeout value is also used for any further request-response pair exchanged during the session.

SPsec Session timeout – the maximum time allowed between any further communication in this session (typically a few seconds).

Upon any timeout detection, the detecting party ends the session.

After a SPsec Session was successfully started, the Configurator may use these services for read and write accesses to parameters (registers) in the Participant.

6.2 SPSec Session Register Read Access

Each transfer starts with an initiation phase where Client and Server agree on the register number and data size to be transferred. The second phase is the data exchange, which can be segmented, if the data does not fit into a single message. The message counter is increment before transmission of a message.

Parameters of ClientReadInitiate():

- mt: Message Type (CPMT_SESS_RDINIT)
- pid: Participant Node ID
- cnt: Shared message counter
- reg: Register number to read
- len: Receive buffer data length
- tag: Authentication tag

Parameters of ServerReadInitiate():

- mt: Message Type (CPMT_SESS_RDINIT)
- pid: Participant Node ID
- cnt: Shared message counter
- reg: Register number to read
- len: Register data length (0, if not readable or register does not exist)
- tag: Authentication tag

If Receive buffer data length and Register data length are identical, then this automatically determines for both Client and Server how many segments will be exchanged. Each segment carries the maximum possible data amount as allowed by the mapping document.

If Receive buffer data length is larger than the Register data length, then the Register data length is used to determine how many segments will be exchanged.

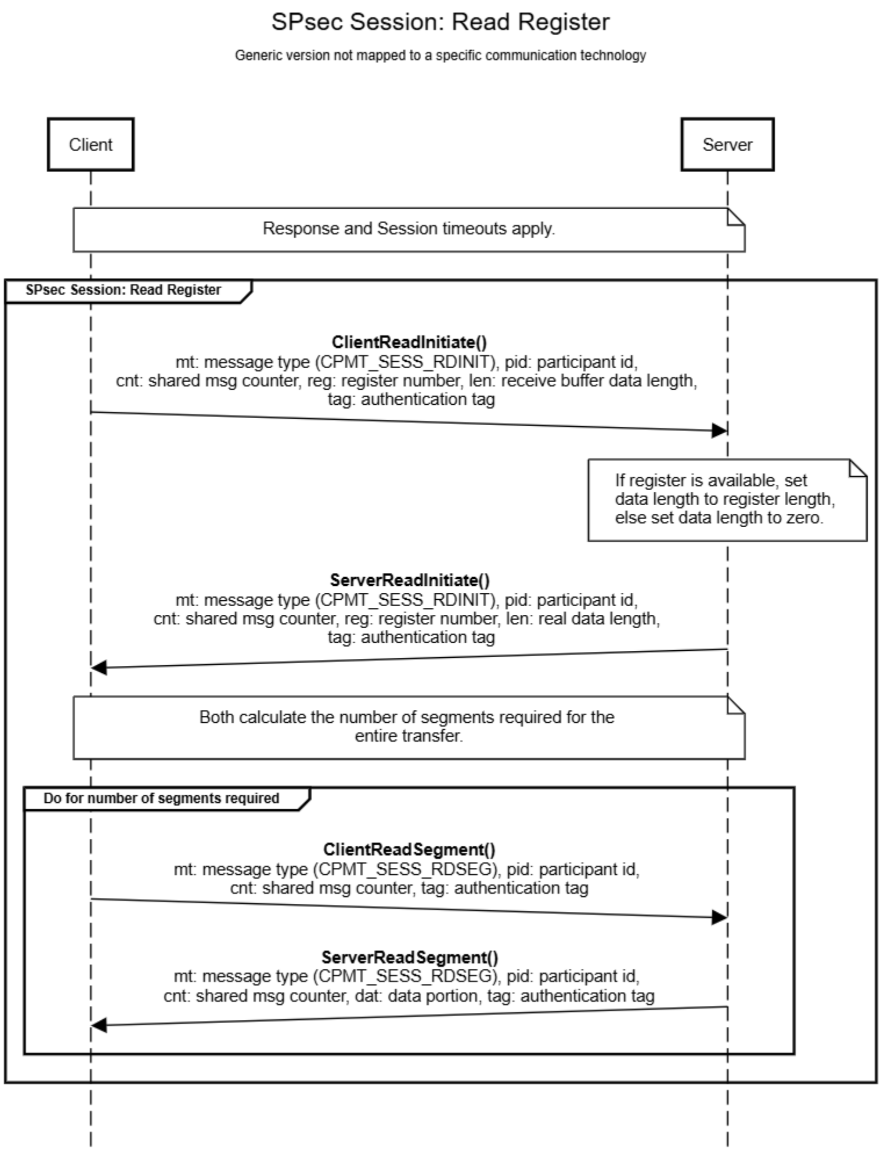
If Receive buffer data length is smaller than the Register data length, then data size is considered zero and no segments will be exchanged.

Parameters of ClientReadSegment():

- mt: Message Type (CPMT_SESS_RDSEG)
- pid: Participant Node ID
- cnt: Shared message counter
- tag: Authentication tag

Parameters of ServerReadSegment():

- mt: Message Type (CPMT_SESS_RDSEG)
- pid: Participant Node ID
- cnt: Shared message counter
- dat: Data segment
- tag: Authentication tag


SPsec Session: Read Register
Generic version not mapped to a specific communication technology
Client
Server
Response and Session timeouts apply.
SPsec Session: Read Register
ClientReadInitiate()
mt: message type (CPMT_SESS_RDINIT), pid: participant id,
cnt: shared msg counter, reg: register number, len: receive buffer data length,
tag: authentication tag
If register is available, set data length to register length, else set data length to zero.
ServerReadInitiate()
mt: message type (CPMT_SESS_RDINIT), pid: participant id,
cnt: shared msg counter, reg: register number, len: real data length,
tag: authentication tag
Both calculate the number of segments required for the entire transfer.
Do for number of segments required
ClientReadSegment()
mt: message type (CPMT_SESS_RDSEG), pid: participant id,
cnt: shared msg counter, tag: authentication tag
ServerReadSegment()
mt: message type (CPMT_SESS_RDSEG), pid: participant id,
cnt: shared msg counter, dat: data portion, tag: authentication tag

The network mapping document defines how the parameters are mapped into addressed data units and an optional segmentation of large register content.

6.3 SPsec Session Register Write Access

Each transfer starts with an initiation phase where Client and Server agree on the register number and data size to be transferred. The second phase is the data exchange, which can be segmented, if the data does not fit into a single message. The message counter is increment before transmission of a message.

Parameters of ClientWriteInitiate():

- mt: Message Type (CPMT_SESS_WRINIT)
- pid: Participant Node ID
- cnt: Shared message counter
- reg: Register number to write
- len: Data length to write
- tag: Authentication tag

Parameters of ServerWriteInitiate():

- mt: Message Type (CPMT_SESS_WRINIT)
- pid: Participant Node ID
- cnt: Shared message counter
- reg: Register number to write
- len: Max data length available (0, if read-only or register does not exists)
- tag: Authentication tag

If Max data length available is equal to or greater than the Data length to write, then Data length to write automatically determines for both Client and Server how many segments will be exchanged. Each segment carries the maximum possible data amount as allowed by the mapping document.

If Max data length available is smaller than Data length to write, then data size is considered zero and no segments will be exchanged.

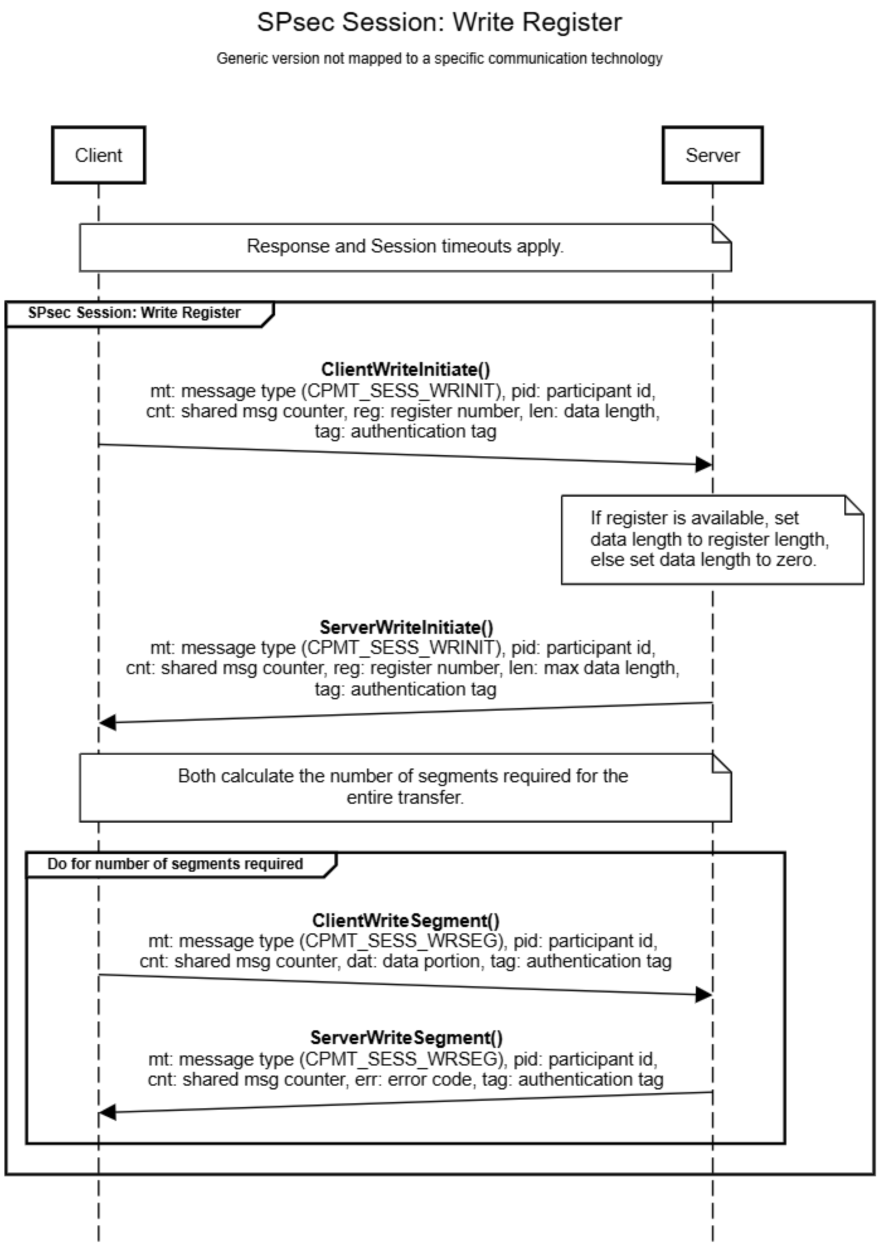
Parameters of ClientWriteSegment():

- mt: Message Type (CPMT_SESS_WRSEG)
- pid: Participant Node ID
- cnt: Shared message counter
- dat: Data segment
- tag: Authentication tag

Parameters of ServerWriteSegment():

- mt: Message Type (CPMT_SESS_WRSEG)
- pid: Participant Node ID
- cnt: Shared message counter

- err: Error code (0 if no error, 1 write error)
- tag: Authentication tag


SPsec Session: Write Register
Generic version not mapped to a specific communication technology
Client
Server
Response and Session timeouts apply.
SPsec Session: Write Register
ClientWriteInitiate()
mt: message type (CPMT_SESS_WRINIT), pid: participant id, cnt: shared msg counter, reg: register number, len: data length, tag: authentication tag
If register is available, set data length to register length, else set data length to zero.
ServerWriteInitiate()
mt: message type (CPMT_SESS_WRINIT), pid: participant id, cnt: shared msg counter, reg: register number, len: max data length, tag: authentication tag
Both calculate the number of segments required for the entire transfer.
Do for number of segments required
ClientWriteSegment()
mt: message type (CPMT_SESS_WRSEG), pid: participant id, cnt: shared msg counter, dat: data portion, tag: authentication tag
ServerWriteSegment()
mt: message type (CPMT_SESS_WRSEG), pid: participant id, cnt: shared msg counter, err: error code, tag: authentication tag

The network mapping document defines how the parameters are mapped into addressed data units.

6.4 SPsec Session Terminate

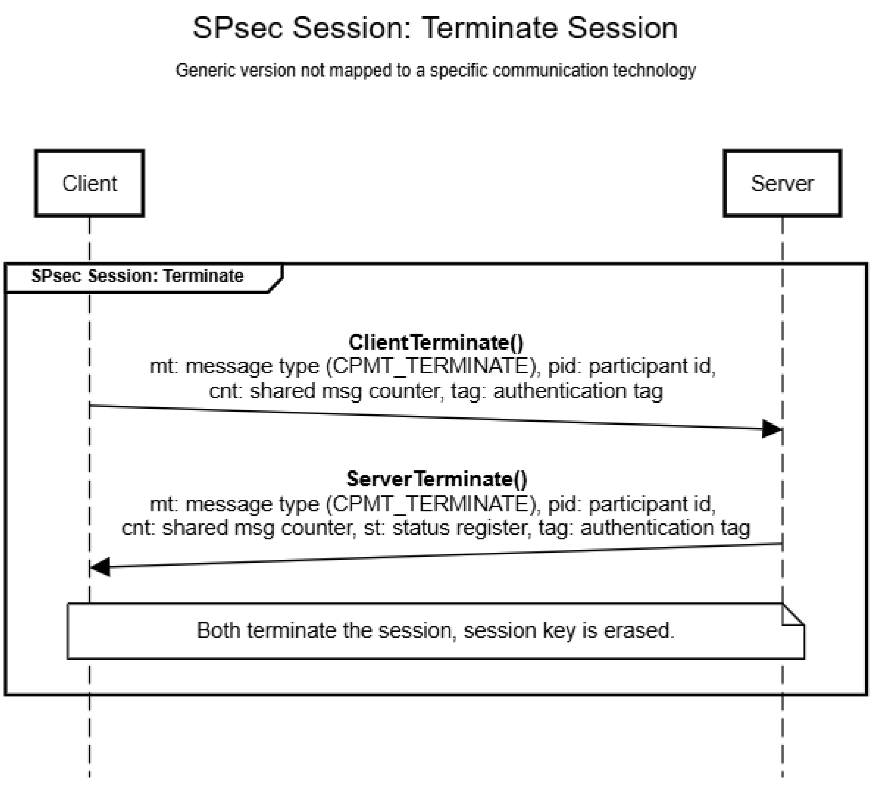
Both the Configurator and the Participant may request an end to an open SPsec Session at any time.

Parameters by Configurator (client):

- mt: Message Type (CPMT_SESS_TERMINATE)
- pid: Particpant Node ID
- cnt: Shared Message Counter
- tag: Authentication tag

Parameters by Participant (server):

- mt: Message Type (CPMT_SESS_TERMINATE)
- pid: Particpant Node ID
- cnt: Shared Message Counter
- st: Value of status register
- tag: Authentication Tag


SPsec Session: Terminate Session
Generic version not mapped to a specific communication technology
Client
Server
SPsec Session: Terminate
ClientTerminate()
mt: message type (CPMT_TERMINATE), pid: participant id,
cnt: shared msg counter, tag: authentication tag
ServerTerminate()
mt: message type (CPMT_TERMINATE), pid: participant id,
cnt: shared msg counter, st: status register, tag: authentication tag
Both terminate the session, session key is erased.

The network mapping document defines how the parameters are mapped into addressed data units and an optional segmentation of large register content.

6.5 SPsec Authenticate Parameter

This service is used to authenticate a parameter such as the time sync vale.

This is done by a variation of the SPsec Session service. Instead of a random IV, the server hello contains the parameter data and both partners generate the current Communication key from that information. The session automatically ends with the “finish” segment from the server, no further services required.

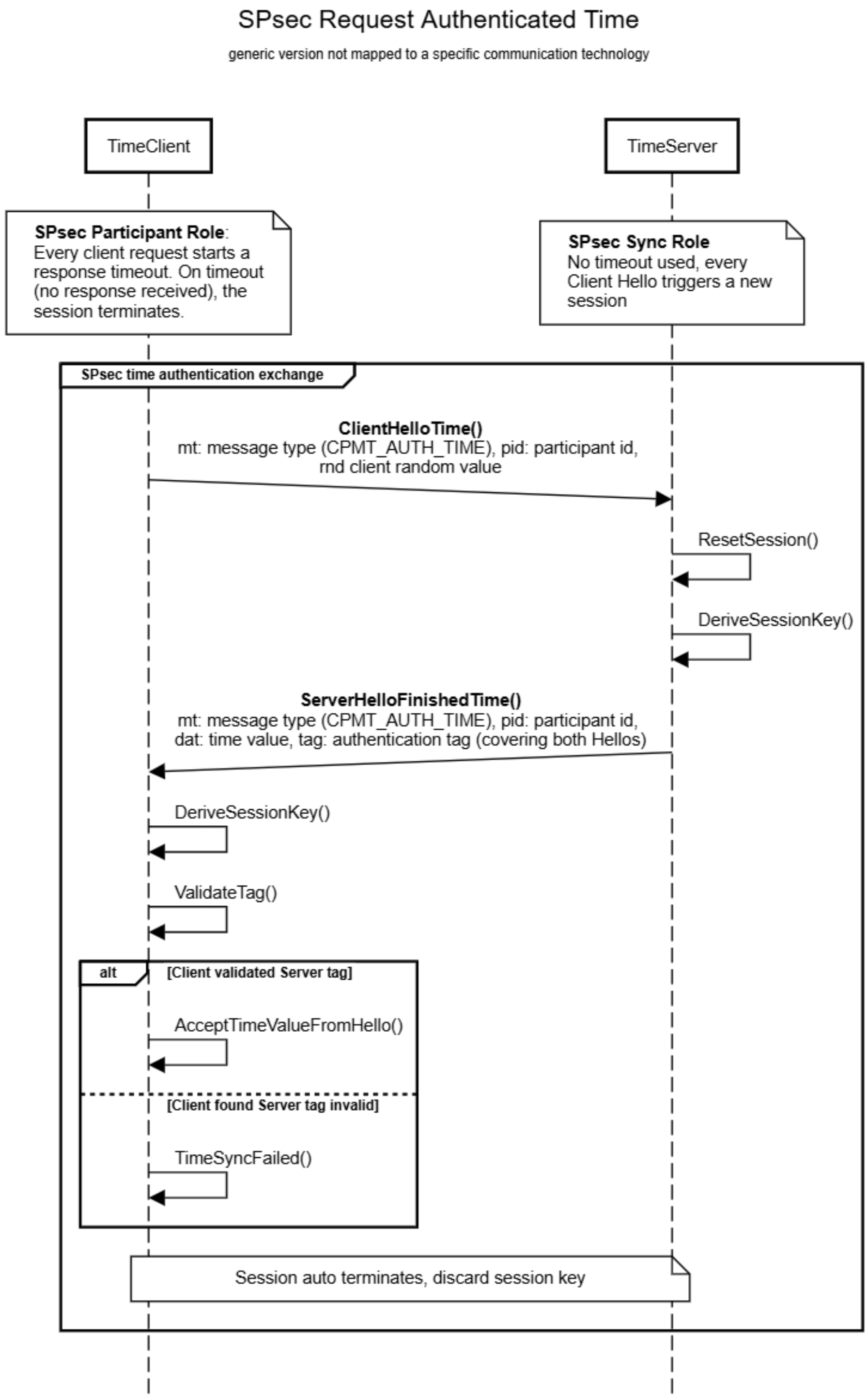
As part of the initialization of the secure communication, every Participant requests the current time information from the Sync role.

Parameters of ClientHelloTime():

- mt: Message Type (CPMT_AUTH)
- pid: Particpant Node ID (source)
- rnd: Random number

Parameters of ServerHelloFinishedTime():

- mt: Message Type (CPMT_AUTH)
- pid: Particpant Node ID (destination)
- dat: Data to authenticate, for example a timestamp
- tag: Authentication Tag (covers both rnd and dat, details in mapping)


SPsec Request Authenticated Time
generic version not mapped to a specific communication technology
TimeClient
TimeServer
SPsec Participant Role: Every client request starts a response timeout. On timeout (no response received), the session terminates.
SPsec Sync Role No timeout used, every Client Hello triggers a new session
SPsec time authentication exchange
ClientHelloTime() mt: message type (CPMT_AUTH_TIME), pid: participant id, rnd client random value
ResetSession()
DeriveSessionKey()
ServerHelloFinishedTime() mt: message type (CPMT_AUTH_TIME), pid: participant id, dat: time value, tag: authentication tag (covering both Hellos)
DeriveSessionKey()
ValidateTag()
alt
[Client validated Server tag]
AcceptTimeValueFromHello()
[Client found Server tag invalid]
TimeSyncFailed()
Session auto terminates, discard session key

The network mapping document defines how the parameters are mapped into addressed data units.

6.6 SPsec Sync Time (for Secure Grouping)

This service is used by a Sync Role to synchronize the timers in all Participants. It is cyclically produced and consumed by all Participants.

It is secured by the same mechanisms that secure data plane communication.

Parameters by Sync Role:

- tim: Current Timestamp
- cor: Correction to previous timestamp (optional)

The network mapping document defines how the parameters are mapped into addressed data units as well as the following time and timeout:

Sync cycle time – the Sync role initiates this service repeatedly based on the Sync cycle time.

Sync timeout – If the service does not repeat within this timeout, then the Sync role is considered lost / removed from the network. By default, this value should be 2.5 times the cycle time.

If the network mapping document supports multiple Sync roles (like a primary and a backup), then corresponding addressed data units need to be defined for all Sync role instances. Furthermore, a mechanism must be present to ensure that the sync value all Sync roles are synced with each other.

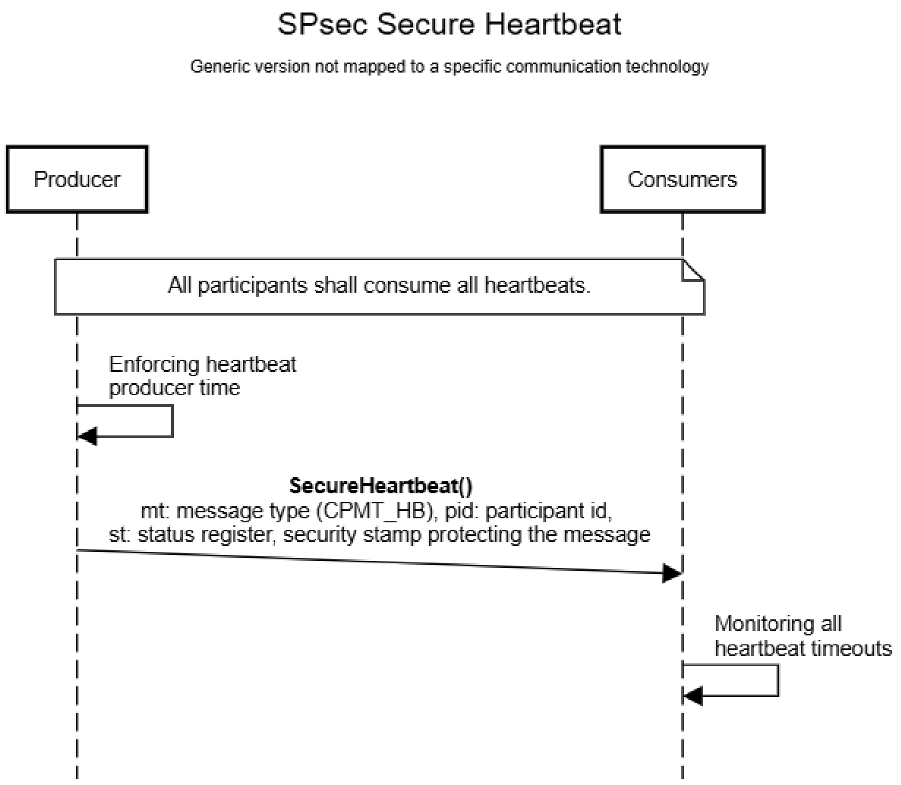
6.7 SPSec Secure Heartbeat

This service is used by all Participants to cyclically share their status.

It is secured by the same mechanisms that secure data plane communication.

Parameters of SecureHeartbeat:

- mt: Message type (CPMT_HB)
- pid: Participant ID
- st: Content of status register


SPsec Secure Heartbeat
Generic version not mapped to a specific communication technology
Producer
Consumers
All participants shall consume all heartbeats.
Enforcing heartbeat producer time
SecureHeartbeat()
mt: message type (CPMT_HB), pid: participant id,
st: status register, security stamp protecting the message
Monitoring all heartbeat timeouts

The network mapping document defines how the parameters are mapped into addressed data units as well as the following time and timeout:

Heartbeat cycle time – the Participants initiate this service repeatedly based on the Heartbeat cycle time.

Heartbeat timeout – If the service does not repeat within this timeout, then the Participant is considered lost / removed from the network. By default, this value should be 2.5 times the cycle time.

The times above shall be the same for all Participants in a network.

For Participants with a synchronized time it is recommended to synchronize the heartbeats such, that they are transmitted in pre-determined time slots based on their node ID.

Example: if the Heartbeat cycle time is 1000ms, Participant 1 transmits its heartbeat on every full second plus 1 millisecond. Participant X transmits its heartbeat on every full second plus X milliseconds.

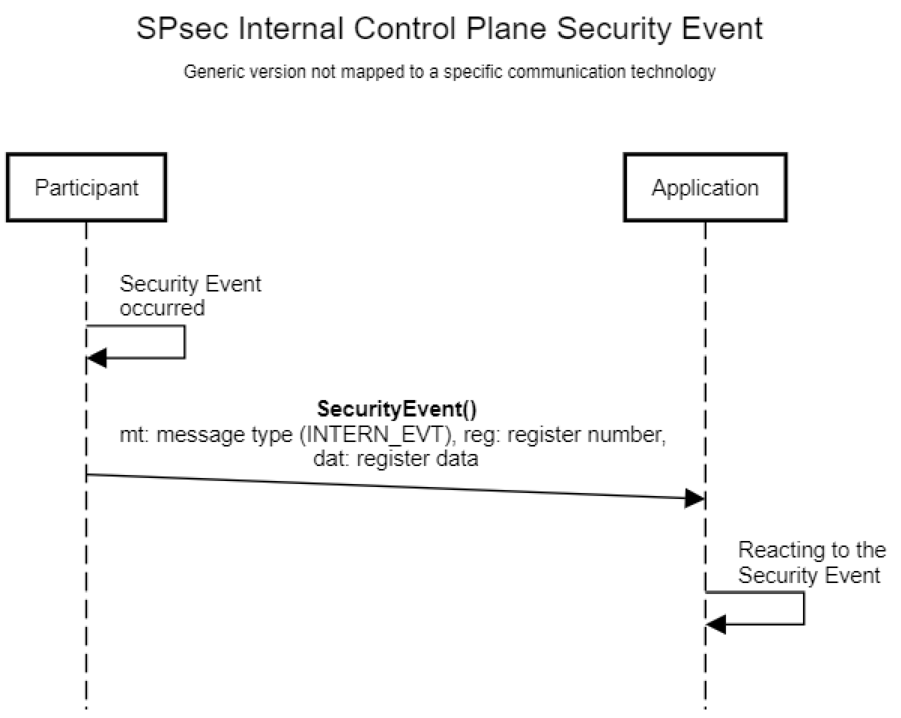
7 Internal Control Plane Services

7.1 Internal SPsec Event

This service is used by SPsec to inform the host about important events, like changes in the FSA status machine.

Parameters:

- mt: Message type (CPMT_INTERN_EVT)
- reg: Register number reporting the event
- dat: Data content of the register


SPsec Internal Control Plane Security Event
Generic version not mapped to a specific communication technology
Participant
Application
Security Event occurred
SecurityEvent()
mt: message type (INTERN_EVT), reg: register number, dat: register data
Reacting to the Security Event

8 Status and Event Indications and Handling

Depending on application and use case it might be desirable or even required to report important changes or events concerning the secure communication. This section summarizes the indications and events that can be reported.

The actions performed upon specific security events are application specific.

In general, a single security event should not trigger an immediate system shutdown as that would have the same consequences as a successful denial-of-service attack. On the other hand, if continuous attacks to a system are recognized, a safe shutdown might be more desirable than a continuous operation under constant attacks.

8.1 LED Security Status Indication

If a device offers status LEDs, the following colours and patterns should be used to indicate the current Security state of the SPsec communication. This can either be done with a single multi colour LED or one green and one red LED.

- SPsec state machine not active (not yet started or shut down):
  Red on, green off
- SPsec state Waiting (for a secure connection)
  Red off, green blinking (200ms on, 800ms off)
- SPsec state Secure (connection active):
  Red off, green on
- SPsec state Warning (that a security event occurred):
  Red and green blinking alternately (400ms alternate),
- SPsec state Configuration (secure configuration active):
  Red off, green blinking (800ms on, 200ms off)

8.2 Security Status and Event Reporting

All SPsec state transitions of the SPsec sub-layer shall be reported to the local host / application. This allows the application to wait until secure communication is available or a configuration is active.

In addition, it is recommended to report the following security events to the host / application. It is up to the application to decide if any of these events are stored in an auditable security event log.

The mapping document defines event codes for the different events below.

The events detectable for secure sessions are:

- Secure session event – Hello, key requested not available
- Secure session event – Finished, key failure
- Secure session event – Server response timeout
- Secure session event – Session timeout
- Secure session event – Key failure during session

The events detectable for timer sync are:

- Time sync event – Authentication request key failure
- Time sync event – Authentication request timeout
- Time sync event – Authentication refresh key failure
- Time sync event – Authentication refresh timeout

The data plane events detectable are:

- Secure data plane event – Authentication failure
- Secure data plane event – Participant heartbeat loss