

SAI MACsec API Proposal

1.0

|  |  |
| --- | --- |
| **Title** | **SAI MACsec API Proposal** |
| **Authors** | **Arista Networks, Inc.** |
| **Status** | **Draft** |
| **Type** |  |
| **Created** | **10/28/2019** |
| **Version** | **1.0** |

**Table of Contents**

[**License**](#_heading=h.1fob9te) **4**

[**Scope & Overview**](#_heading=h.g99jx3ncxpjv) **5**

[Block Diagram](#_heading=h.t2p14zw6bbb) **5**

[**Object Relationships**](#_heading=h.4d34og8) **6**

[**Function call sequences**](#_heading=h.2s8eyo1) **10**

[Enable MACsec on a port](#_heading=h.nufqtif52c5r) 10

[Add a SC with 1 SA for a port](#_heading=h.mddh2zpidngg) 10

[Replace egress SA](#_heading=h.vbeg2ot4bd9p) 10

[Replace ingress SA](#_heading=h.35pcg5uxmut6) 10

[Remove a SC](#_heading=h.rbrglmh8d3ke) 11

[Disable MACsec on a port](#_heading=h.alnmk1hfn10s) 11

[Periodic activities](#_heading=h.p1nlr9yy22y5) 11

[**Notes**](#_heading=h.12r7gxk51c67) **12**

[Unsupported Features](#_heading=h.erj6mj3r76mo) 12

[Hidden Complexities](#_heading=h.2ag6v5yq9l4e) 12

# License

Contributions to this Specification are made under the terms and conditions set forth in Open Web Foundation Contributor License Agreement (“OWF CLA 1.0”) (“Contribution License”) by:

**Arista Networks, Inc**

 Usage of this Specification is governed by the terms and conditions set forth in the Open Web Foundation Final Specification Agreement (“OWFa 1.0”).

**Note**: The following clarifications, which distinguish technology licensed in the Contribution License and/or Specification License from those technologies merely referenced (but not licensed), were accepted by the Incubation Committee of the OCP:

**None**

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Scope & Overview

Scope:

This document defines the technical specifications for the API used to support 802.1ae MAC Security (MACsec) in Open Compute Project Switch Abstraction Interface (SAI). MAC Security Key Agreement Entity (802.1ae KaY), responsible for obtaining AES key and other parameters, is outside the scope of this document and the MACsec API.

Overview:

The switching hardware consists of network interfaces connected to a forwarding element, such as a switching ASIC. Some switching hardware also include Phy ASIC(s) that interconnect network interfaces and forwarding element interfaces. Each Phy ASIC supports one or more network interfaces. SAI is used for each such Phy ASIC (doc/macsec-gearbox/SAI\_Gearbox\_API\_Proposal-v1.0.docx).

This SAI MACsec API is used to control Macsec hardware implemented inside such a Phy ASIC. The SAI MACSec API provides a software interface to a single Phy ASIC that provides hardware support for 802.1ae MACSec Entity (SecY).

MACsec provides data security (confidentiality, integrity, authenticity) to Ethernet port(s). MACsec uses 2 distinct datapath modules:

1. The egress module adds SecTag, encrypts packets and adds a trailer containing ICV (Integrity Check Value).
2. The ingress module decrypts the packet, checks the authentication and integrity (using ICV) and removes the SecTag.

# Block Diagram

The Phy chip contains MACsec functionality that services a set of network ports (a.k.a. Phy line-side ports). Some resources (e.g. encryption keys) are not shared between network ports. Some other resources are shared between multiple network ports e.g. function like parsing, match-criteria for control-packets, etc.



# Object Relationships

The SAI MACsec API provides an interface to programming the MACsec functions in a MACsec-enabled Phy:

1. Two sai\_macsec objects (one for ingress and one for egress) are bound to a sai\_switch object with SAI\_SWITCH\_ATTR\_TYPE == SAI\_SWITCH\_TYPE\_PHY.
2. Two sai\_macsec\_port objects (one for ingress and one for egress) are bound to the line-side port (SAI\_PORT\_CONNECTOR\_ATTR\_LINE\_SIDE\_PORT\_ID) of a sai\_switch object with SAI\_SWITCH\_ATTR\_TYPE == SAI\_SWITCH\_TYPE\_PHY.



The MACsec logic classifies control packets which does not undergo normal security processing. These may include some user-defined frames based on macsec\_port, dst\_mac and ethertype (after vlan tags) or some combination of those.

The SAI MACsec API supports non-control packets with or without a vlan tag. It can classify flows based on (macsec\_port\_id, dst\_mac, ethertype) or a subset of these to identify the MACsec Secure Channel (SC). Some implementations may be capable to classify based on outermost vlan tag values for egress and, if VLAN is not encrypted, for ingress. The ingress also uses SCI and AN fields from the 802.1ae SecTag header to identify the SC and Security Association (SA).

MACsec processing for all other packets is based on flow identification and associated action. Flow action can be of 4 types:

1. Deny i.e. drop packet
2. Permit i.e. send packet without MACsec encryption and count as non-control packets
3. Control\_packet i.e. permitted and counted as control packets
4. Secure

A flow can be associated with:

1. One or more macsec\_rule
2. One or more IEEE 802.1ae SCs, used only with flows which have Secure flow-action.

Flow-rules are used to match with macsec\_port\_id and packet header fields to identify a flow. Each macsec\_port has a table of rules (analogous to SAI ACL table). Each rule is assigned a unique ordinal priority value within the scope of a table.



For a MACsec-disabled port, all packets should be permitted. For a MACsec-enabled port, normally the flow rules have the following precedence order:

1. permit entries for control packets
2. secure entries for one or more SC
3. deny entry for all other packets



A SC has one or more associated SAs. Each SA has configuration parameters for its cipher suite (encryption key, etc). Each encrypted packet has a packet number (32-bit PN or 64-bit XPN). For each SA, the egress uses a counter to increment the packet number. During the lifetime of a SA, the packet number value used by the egress is periodically sent to the receiver(s) by the key agreement protocol. This ingress logic uses this value as the minimum acceptable value of packet number. The SA is periodically replaced before it exhausts all its packet numbers. The SC can operate uninterrupted for a long time, using rekeying i.e. using a series of overlapping SAs.

1. For egress, only one SA is active at a time. Rekeying involves switching the active SA and changing the SecTAG.AN field in the packet header.
2. For an ingress secure packet flow, multiple SAs can be simultaneously be active, selected for each packet by SecTAG.AN field from the packet header.

For ingress, the SecTAG.SCI field from the packet header has to be matched to a programmed value. This API supports 2 implementation models:

1. If SCI is not matched as part of a macsec\_rule, multiple SCs (e.g. SC-21 and SC-22 in Figure 4) can be associated with a flow (e.g. Secure flow-2). Then a specific value of SCI from the SecTAG in the packet is used to identify a specific SC associated with a flow.
2. If SCI is matched as part of a macsec\_rule, only one flow-rule and one SC is associated with each flow.

# Function call sequences

The following sequences assume that a MACsec object associated with the Phy has already been created by a create\_macsec function call.

## Enable MACsec on a port

// Step-1: Create a MACsec port

create\_macsec\_port

// Step-2: Set up and enable control packet processing for this port

create\_macsec\_flow // with action = control packet

create\_macsec\_rule // with high priority and matching control packet ethertype

set\_macsec\_rule\_attribute // enable rule

// Step-3: Set up and enable default action for MACsec data packet for this port

create\_macsec\_flow // typical default action = deny

create\_macsec\_rule // with lowest priority and matching all packets

set\_macsec\_rule\_attribute // enable rule

## Add a SC with 1 SA for a port

// enable data-packet flow and rules for this port and then enable MACSEC

create\_macsec\_flow // action = secure

create\_macsec\_rule // with medium priority and desired match criteria

// add more rules, if needed

// After getting MACsec SC/SA parameters from key agreement entity

create\_macsec\_secure\_channel

create\_macsec\_security\_association // 1st SA for this SC

set\_macsec\_rule\_attribute // iterate to enable all rules for this SC

## Replace egress SA

// After getting parameters for another SA from key agreement entity

create\_macsec\_security\_association

// When it is time to change the SA

set\_macsec\_secure\_channel\_attribute // Change active SA

get\_macsec\_security\_association\_stats\_ext // collect last statistics for old SA

remove\_macsec\_security\_association // remove old SA

## Replace ingress SA

// After getting parameters for another SA from key agreement entity

create\_macsec\_security\_association

// 0.5 sec after receiving a packet with new SA

get\_macsec\_security\_association\_stats\_ext // collect last statistics for old SA

sai\_remove\_macsec\_security\_association

## Remove a SC

set\_macsec\_rule\_attribute // iterate to disable all rules for this SC

get\_macsec\_security\_association\_stats\_ext // iterate over all SA for this SC

sai\_remove\_macsec\_security\_association // iterate over all SAs for this SC

get\_macsec\_secure\_channel\_stats\_ext // collect last statistics

sai\_remove\_macsec\_secure\_channel

remove\_macsec\_rule // iterate to remove all rules for this SC

get\_macsec\_flow\_stats\_ext // collect last statistics

sai\_remove\_macsec\_flow

## Disable MACsec on a port

// Step1: Disable & remove default action for MACsec data packet for this port

set\_macsec\_rule\_attribute // disable default data rule

remove\_macsec\_rule // default data rule

get\_macsec\_flow\_stats\_ext // collect last MACsec default data statistics

remove\_macsec\_flow // default data flow

// Step-2: Disable & remove control packet processing for this port

set\_macsec\_rule\_attribute // disable control packet rule

remove\_macsec\_rule // control packet rule

get\_macsec\_flow\_stats\_ext // collect last control packet statistics

remove\_macsec\_flow // control packet flow

// Step3:

remove\_macsec\_port

## Periodic activities

1. Read statistics
2. Read egress packet number, SAI\_MACSEC\_SECURITY\_ASSOCIATION\_ATTR\_XPN. The Key Agreement Entity sends this value to the MACsec peer.
3. Set minimum acceptable packet number, SAI\_MACSEC\_SECURITY\_ASSOCIATION\_ATTR\_MINIMUM\_XPN, for ingress. The Key Agreement Entity sets this to the value provided by MACsec peer.

# Notes

## Unsupported Features

1. Packet parsing beyond MAC header and at most 2 VLAN tags.
2. The SAI MACsec API does not support any interrupt or event notification to NOS. Possible uses of notification could include:
   1. Egress packet number crossing threshold. The last used egress packet number can be polled or indirectly inferred from periodically collected egress packet statistics.
   2. A new packet SA value observed at ingress. This can be polled to determine when to remove the old SA.
   3. External Dependencies

The MACsec API makes use of AES key and other parameters obtained by MAC Security Key Agreement Entity (802.1ae KaY). It is the responsibility of the application (NOS or Controller) to implement the MAC Security Key Agreement Entity (802.1ae KaY).

## Hidden Complexities

Some implementation complexities are hidden to simplify the SAI MACsec API:

1. The SAI MACsec API exposes rules as a prioritized list of rules per port. But the driver flattens the general-purpose rules for all ports into a single priority space for a single TCAM in hardware implementation. So this API shows a single pool of rule resource for all the ports of a Phy.
2. The SAI MACsec API reports the availability of general-purpose flow and rule resources. But that does not include specialized hardware resources used to support control packets and default action for data packets. The driver may use rule match-criteria and flow-action to identify those which can be mapped to such specialized hardware. For such rules and flows, this API may not show any change in resource availability, and the control packet and default action for data packets may appear to "come for free".