Preemption and MACsec replay protection

Mick Seaman

Use of the proposed IEEE 802.3 Ethernet frame preemption capability could result in frame reordering. Without a change in the MACsec specification (IEEE Std 802.1AE) or in the way it is used it would not be possible to use MACsec to provide strict replay protection.

This note has been revised to detail the use of multiple SCs[1] per port, following the 802.1 Security Task Group discussions[2] at the November 2014 meeting. This is the preferred approach to maintaining the current replay protection capabilities because it also addresses traffic class based reordering by Ethernet Virtual Circuits and similar provider network services[3]. The discussion of alternative approaches has been retained (somewhat reordered) for the record.

The use of multiple SCs by a single MKA participant requires some changes to 802.1X[4] and to existing MKA implementations. This note points out just how simple these can be. An existing and unchanged conformant implementation of MACsec/MKA should interoperate with one that uses traffic class grouping SCs.

1. Introduction

Before detailing mechanisms and ‘solutions’, this note reviews:

a) MACsec’s threat model (2.1) and goals (2.2)

b) How frames can be misordered by preemption (2.3) and provider network services (2.4)

Preemption is only the latest of a number of MAC-specific mechanisms for differential services supporting various notions of ‘priority’, ‘timeliness’, and bandwidth sharing. This note describes:

c) The generalized model of ‘priority’ handling previously developed in 802.1, and its applicability to our present issues (2.5)

It also provides MACsec and MKA (MACsec Key Agreement) background essential to understanding the details of the proposed multiple SC approach and the alternatives, describing:

d) The CAs (Secure Connectivity Associations), SCs (Secure Channels) and SAs (Secure Associations) that protect data frames (2.6).

e) How and why MACsec provides replay protection (2.7)

f) Details of MKA operation (2.8) and MACsec frame protection and validation (2.9)

Those familiar with all of the above might want to skip straight to the proposed use of multiple SCs. In currently approved standards[5], a single MACsec-capable real[6] port incorporates a SecY that uses a single transmit SC[7] controlled by a single KaY[8]. To support preemption and strict replay/misordering protection, that SecY would use, and the KaY would control, two SCs, one transmitting and receiving frames for *express*[9] traffic classes and the other for *preemptable* traffic classes. To support the use of differentiated services and strict replay/misordering protection over a provider network, the SecY would use two or more SCIs, each

[1] ‘Multiple (per traffic class group) SCs’ is a more accurate description of what was described in the meeting as ’multiple SAs’. Each SC is already supported by an overlapping sequence of SAs.

[2] I am indebted to Brian Weis’ notes of our discussions, and his contributions .../docs2014/ae-weis-preemption-1030-v00.pdf and .../docs2014/ae-weis-replay-protection-and-preemption-1105-v00.pdf. The MKA and other changes needed to support traffic class SAs, as described in this note, are simpler than discussed in those contributions.

[3] Making the current P802.1AEcg project an appropriate place to standardize the solution. A very brief discussion and reference should be included in P802.1Qbu.

[4] The necessary changes to 802.1X, detailed here, are slight and might be held over to a future amendment or revision with the discussion in P802.1AEcg proving sufficient in the interim.

[5] Including 802.1X-2010 and 802.1AE-2006.

[6] ‘Real port’ as opposed to ‘virtual port’ as described in some detail in 802.1AE-2006 clause 11.8 and in 802.1X-2010 clause 3, 5.12, 5.21, 6.3.6, 7.5, 12.1, 12.7, 12.9 and elsewhere. This note does not describe virtual ports in detail, the concepts that are discussed are readily extensible to virtual ports.

[7] SecY: MAC Security Entity, i.e. the entity that actually uses MACsec to protect and validates data frames. SC: The (one-way) Secure Channel used by each SecY to transmit to its peers.

[8] KaY: Key Agreement Entity i.e. the entity that participates in MKA to create and control the SCs and SAs used by MACsec.

[9] That is frames that can *preempt* preemptable frames that are in the process of being transmitted.

transmitting and receiving frames for the set of traffic classes that map to one of the provider's service classes[10]. This note describes:

g) The basic model, with two (or more) transmit SCs each using a distinct SCI just as if they were supporting distinct ports in a single group CA, (3.1)

h) MKA operation and efficient implementation for multiple SC/SCI support (3.2)

i) How the existing SecY MAC and PAE[11] MIBs are used, and what extensions are necessary (3.3)

A number of alternative approaches to addressing the dual problems of providing replay and misordering protection over links that support preemption, and over provider network services, have been discussed and are included in this note for the record:

j) No change to the current MACsec specification (4.1).

k) 'Scoreboarding', keeping a complete record of recently received PNs (4.2).

l) Relying only on the existing preliminary recheck, removing the check performed when a received frame has been validated (4.3).

m)Revising the MACSec packet format to separate the PN's cryptographic nonce and replay protection functions, replacing the later with an additional sequence number carried at the end of the protected frame, immediately prior to the ICV. This alternative approach is only mentionned for completeness, least it be suggested at a late stage in the standards process, and can be considered and rejected now.

The last of these alternatives ((l) and (m) above) apply only to preemption on a single link, and not to provider network services that may traverse multiple links (any one of which might reorder frames by using differential queueing or indeed preemption) below the level of the MAcsec hop.

2. Background

2.1 Threat model

MACsec assumes that an attacker can copy, modify, remove, and add frames at will.

2.2 MACsec goals

IEEE 802.1AE details MACsec's goals, but for our present purposes a somewhat higher level view is useful: in addition to addressing issues of confidentiality, MACsec attempts to avoid the need for each and every higher level protocol designer to craft protocol-specific mechanisms to counter attacks using the posited threats (2.1. above). MACsec operates in the context of a single LAN[12] and of perimeter security, to reduce or remove effects that attacks made on that LAN can have on the operation of the rest of the network. An attacker possessing access to a LAN and the capabilities assumed in the threat model can clearly make that LAN unusable, but MACsec ensures that it is localized and hence easier to contain, investigate, and counter.

2.3 Preemption and misordering

Preemption in the Ethernet MAC, as currently proposed[13], allows a frame that is currently being transmitted to be interrupted by one or more preempting frames. Once the prempting frames have been transmitted, in their entirety, transmission of the preempted frame resumes (without any retransmission of the data already sent) though it may be prempted once more. Preempting frames cannot themselves be prempted, nor can a frame that has the attribute of being able to preempt another if it happened to be transmitted while the latter was still 'on the wire'[14]. There is a lower limit to the size of fragments that a preempted frame can be broken into, of the order of 64 octets, with the exception of final fragment[15]. At present there is no bound on the number of preempting frames that can be sent.

If there are no intervening bridges, two commmunicating systems experience misordering as a consequence of the complete reception of a preempting frame occuring before the complete reception of the frame it preempts. The initial octets of both frames are received in order.

However if there is an intervening bridge, possibly operating transparently to the attached systems (as would a TPMR between two Customer Bridges) then that would receive and reorder the frames before transmitting them. Thus the receiving system might

[10] Since the network provider could well be using provider bridges and provider backbone bridges the same terms might be used to refer both to the customer's use of priority and traffic classes and to the methods used by the provider to support the differentiated services offered. To avoid lengthy and repetitive clarification this note reserves the terms 'differentiated services', 'service classes', and 'service priority' to descrive provider network operation.

[11] A PAE (Port Access Entity) implements 802.1X including MKA in a Supplicant or an Authenticator.

[12] Though that LAN can be large and complex, see MACsec hops http://www.ieee802.org/1/files/public/docs2013/ae-seaman-macsec-hops-0626-v03.pdf

[13] So far as I understand it.

[14]

[15] This needs checking, it would be nice from an implementation performance point of view if the minimum size of the final fragment was also constrained.

receive the two frames in their entirety, without one seeming to preempt the other. The attribute ‘preempting’ is not, therefore, firmly associated with a transmitted frame and cannot be used as part of a general solution to the problem of retaining strict replay protection capability when preemption is being used. The TPMR or even more invisible device might have been inserted by an attacker.

On the other hand preemption is of use in scenarios where timing and resource allocation are tightly controlled and is probably of marginal utility if there are intervening buffering systems that cannot know (because MACsec is rendering the contents of each frame confidential) which frames are candidates for preemption or being preempted. So we do not neccessarily need a solution that will not discard otherwise good frames because they have passed through some intermediate low-level bridge.

2.4 Misordering in provider networks

MACsec operates over a single hop and in most cases that corresponds to a single physical LAN. However where a virtual LAN service is supported by a provider network the ‘hop’ can be provided by a number of bridges that can reorder frames based on requested priority. The issues that arise are discussed in .../docs2013/ae-seaman-macsec-hops-0626-v03.pdf .../docs2013/ae-seaman-ede-0713-v02.pdf and 802-1-AEcg-d0-3.pdf.

2.5 Priority in a layered architecture

IEEE 802.4 (Token Bus) and IEEE 802.5 (Token Ring) both provided up to eight levels of priority. These priorities were both signalled to peer service users and used to control access to the LAN medium, in fairly sophisticated ways that provided for bandwidth sharing as well as strict prioritization. A desirable level of flexibility was provided by not tying the signalled values (user_priority) and control over access to the medium (access_priority) together, but providing both as parameters of MAC Service requests. IEEE 802.1D-2004 and IEEE 802.1Q-2003[16] also provided them as parameters as the (E)ISS. However, in developing 802.1ad and 802.1Q-2005, we found that this approach, which could be summarized as requesting both what the user wants (user_priority) and how to use local mechanisms to support that request (access_priority), does not work well when services are layered or supported by number of concatenated lower layer hops. It supposes that each sub-layer knows not only what it wants the service (of the supporting sub-layer) that it is using to communicate to its peer users, but also how that lower sublayer should do its job. It is itself being told (by its user) exactly how to do its job, omitting the small detail of what it should ask of its supporting sub-layer.

From 2005 we adopted a simpler model, suitable for (sub-)layering. Each request carries a single priority value which the (sub-)layer both communicates (possibly modified by the concatenation of sub-layer services) to its peer and uses (in the station where the request is made, and at relaying stations enroute to the peer) to select the appropriate actions, including the parameters of any requests to further supporting sub-layers.

This experience is valuable in the present case of pre-emption. Much as it might seem that pre-empting services can be modelled by the operation of separate ports within a supporting sub-layer, that conclusion is particular to the internal operation of the sub-layer and does not affect its need to receive and interpret the requested user priority and to use that to determine the access priority of any further sublayer that it uses.

2.6 CAs, SCs, and SAs

For a detailed description of these constructs see 802.1AE-2006 clause 7.1 and Figures 7-1 through 7-6. A CA, the Secure Connectivity Association that secures the MAC Service between a set of peer ports[17], is supported by a number of SCs (secure channels) each of which allows a CA member to issue a single service request to transmit a frame to all the other CA members[18]. Each SC is supported by a succession of overlapping SAs (secure associations), with the change from each SA to its successor being necessitated by the use of a fresh symmetric key (SAK) for reasons of policy or key exhaustion[19].

[16] And earlier editions of those standards.

[17] Here ‘port’ is a short-hand for the general concept of ‘service access point’, noit a reference to a physical plug. See 802.1X-2010 for an introduction to basic architectural concepts and terms.

[18] Note the one-way nature of an SC, and the fact that group addresses and individually addressed frames are not handled differently. In a bridged network, where individually addressed frames can be be broadcast/multicast if their destination has not been previously learnt or configured, individually addressed frames are only a potentially filterable case of broadcast, as are multicast frames. The way in which a frame is transmitted is not changed if some component of an underlying service has increased its efficiency by reducing the set of paths it is to traverse to reach a possible destination.

[19] Each secure assocation symmetric key (SAK) requires the use of a unique initial value (IV) for every frame transmitted with that key, so SAKs have to be changed before the IV is forced to repeat itself, will it will eventually do if the part of the IV that provides uniqueness is an incrementing packet number (PN) in a finite field, as is the case with MACsec.

2.7 MACsec replay protection

MACsec adds a SecTAG to each frame to carry, amongst other parameters, a packet number (PN) that is incremented with each frame transmitted and provides each instantiation of the symmetric cryptography primitives with the unique nonce it requires. On receipt MACsec provides a configurable replay window, the leading edge of which is determined by the PNs of successfully validated frames. If the replay window is zero then a subsequent frame is only accepted if its PN is greater. In that case MACsec provides strict replay protection. Otherwise frames within the trailing window are accepted, even if repeated.

Some protocol stacks behave logically correctly but with significantly reduced performance if frames are received out of order. For example, end station implementations of TCP/IP used to (and may still) suffer in this way. A misordering attack thus has at least some nuisance value at a distance[20].

Other protocol entities are more dependent on in-order reception. For example, simple registration protocols with idempotent messages generally assume that the state of each entity's peers is contained in the last received message. Of course replay protection cannot defeat attacks that could be equally made by simply removing frames from the communication—simple lack of registration for example. The additional risk that replay protection can guard against is that of a flapping attack—alternating repetition of old and new messages that cause the receiving MACsec protected entity to propagate alternating states to distant entities.

MACsec can protect against the indefinite repetition of such messages by bounding the transit delay of protected frames. MKA (MACsec Key Agreement) carries the necessary PN information to advance the lower edge of the replayWindow if the transmission rate itself is not sufficient to guard against replay. The periodic transmissions used by simple protocols to guarantee convergence after loss thus suffice to limit the time during which flapping attacks can be made: when registrations (or similar demands placed on the network) are stable there are no old messages that are sufficiently recent to be accepted by MACsec.

2.8 MKA operation

For the details of MKA operation see IEEE 802.1X-2010, as amended by P802.1Xbx D1.6[21].

2.9 MACsec protection and validation

2.9.1 Receive validation

Details of MACsec's receive processing are modelled as shown in Figure 1 (a copy of Figure 10-5 of 802.1AE-2006 as amended by 802.1AEbw-2013). The standard does not assume that all valid implementations can carry out cryptographic validation of receive frames irrespective of LAN utilization and frame size, however desirable that might be, and thus its model[22] of receive processing includes a receive fifo (shown halfway up the figure) prior to validation. A number of operations can or need to be performed prior validation and there was no reason to model those as being implemented at other than full line rate. They also, conveniently for the purposes of the current discussion, involve no frame data other than that present in the SecTAG and thus present in the initial octets of any received frame, whether pre-empted or not[23]. Contrariwise frame validation, and hence the operations shown in the upper half of the figure, cannot be completed until a entire frame has been received. The receive model can therefore be taken (if desired) as applying to a stream of prempting and preempted frames with in order receipt in the bottom half of the figure followed by some reordering in the fifo as preempting frames overtake their immediate preempted predecessor.

Replay protection is modelled as occuring both before and after frame validation, that is to say both in the lower half of the figure before the receive fifo and in the upper half after validation. The lowest acceptable PN can only be updated by a received frame (to the value of the PN carried plus one minus the size of the replay window[24]) after that frame has been successfully validated. If the replay window size is to

[20] I use 'at a distance' in contrast to 'localized' to mean an attack that can have a network effect that extends beyond the receiving MAC Security Entity.

[21] Currently awaiting formal approval by IEEE RevCom. Time permitting this section will be expanded as advertised in the Introduction of this note.

[22] As always the standard points out that its model of operation is simply a basis for describing functionality and real implementations may adopt any internal model of operation compatible with the externally visible behavior that the standard specifies.

[23] The one test where this might be in doubt is that labelled in the Figure 'if(invalid_tag_or_icv(rx)'. At this point the only test applied to the ICV is to ensure that the received frame is of sufficient length to contain a correctly formatted SecTAG and a ICV of corresponding length. If MACsec is permitted to make the initial fragment of any premepted or preemptible frame at least 80 octets long (unless the entire frame is shorter) then this test can remain unchanged. Otherwise the test for ICV presence and size could be delayed until the frame is to be validated, after the fifo discussed. The only externally visible change resulting from such a change would be the incrementing of one error count rather than another if a frame is both too short to contain the specified ICV and in error so far as another test to be performed prior to the fifo is concerned.

[24] Obviously the lowest acceptable PN is not updated if the received value is, while acceptable, so low as to lower the value of the lowest acceptable.

be accurately enforced, the PN of each frame has to be checked against a lowest acceptable PN that could have been updated by the immediately prior frame.

2.9.2 Transmit protection

Details of MACsec's transmit processing are modelled as shown in Figure 2. As with receive processing, the standard does not assume that all valid implementations can carry out cryptographic protection at line rate and the model includes a transmit fifo. It matches the rate at which Controlled Port service requests occur, which could be arbitrary, with protection and line rates. This fifo is arguably unneccessary: service primitives are observed events not procedure calls or other local interface operations —if the MACsec Controlled Port does not accept a frame from its service user the corresponding request primitive does not happen—the buffering is forced to exist elsewhere in the model of the overall system. However, when we come to modelling processing at a finer granularity than that of complete frames, buffering might play a role.

The PN acts as a nonce for the symmetric cryptography used in the protection operation, and is thus assigned as soon as the cipher suite[25] starts to calculate, AES block by AES block, the octets of a frame that will be transmitted and the integrity check value (ICV) that will be appended to the frame. The cipher suite currently standardized for use with .1AE use Galois Counter Mode (GCM) which has the advantage that the basic operations that it uses can be performed in parallel at Ethernet line rates[26]. This makes low latency implementations possible—octets of earlier blocks can be transmitted while those later in the frame are yet to be calculated. Such implementations have also been designed to exhibit constant latency, irrespective of frame size mix, thus supporting the use of .1AS/IEEE-1588.

To support strict replay protection, as specified by current standards, the MACsec transmit processing has to transmit frames in the order that it assigns PNs to those frames, and naturally computes the octets of each transmitted protected frame in that order. This is even true if the MACsec implementation is not tightly coupled to the Ethernet MAC but completes its calculation of the octets of the protected frame to be transmitted before submitting any of them for transmission. One could imagine an implementation of a bridge using the GCM and AES instructions now available on a general purpose processor, part way through the protection computation for a long frame and receiving a short high priority frame. The protection operation on the long frame could be suspended in favour of working on and transmitiing the short frame first. Thus the frames would be reordered in the transmit fifo, even in the absence of preemption capability in the Ethernet MAC. As previously mentionned, .1AE attempts to be reasonably tolerant of implementation diversity. The maximum permitted MACsec latency and jitter allow for processing a maximum sized frame and four minimum sized frames, and thus accomodate the loosely coupled implementation described. However it is unlikely that an implementation with such latency and jitter would meet the timing requirements of the systems that are the intended beneficiaries of preemption capability in the MAC. The latter are highly likely to use tightly coupled implementations that offer minimal jitter and a latency that is somewhat smaller than that corresponding to the transmission of a maximum sized frame. In any case the current .1AE standard does not provide reordering between the assignment of a PN and transmission of the corresponding frame, so (in the absence of intermediate bridges between transmitter and receiver) the initial octets of each frame should be received in transmission order even with preemption, though receipt of the complete frames may be misordered (see 2.9.2. above).

3. Multiple traffic class group SCs

3.1 Development of the basic model

The way in which traffic class grouping SCs are modelled applies equally to point-to-point CAs (with only only two real ports as members of the CA), group CAs (with three or more real ports as members), and multi-access LANs[27] (where the members of each CA comprise one real port and one virtual port). This note focuses on the first of these, as it is likely to:

—be the most common,

—raise more questions than the second as it confers group CA attributes on a point-to-point relationship,

[25] Each cipher suite specifies how the 16-octet block oriented AES is used to encode a longer sequence of octets and how that sequence is composed from the fields of the SecTag and the original frame data.

[26] Implementations processing at 100 Gb/s or more were reported some years ago.

[27] See IEEE 802.1AE-2006 clause 11.8.

— answer the major points that arise in the last and most complex case.

The first major point is that the MACsec protected data frames transmitted for each of the traffic class groups need to be distinguished, securely, from frames transmitted by the others. That is to say they have to be distinguished by a frame field or fields included in the MACsec integrity check. That field or fields should not be subject to confidentiality protection, so a receiver can continue to make some decisions before cryptographic validation (as in Figure 1). That leaves us with few choices, and if existing fields are not to be redefined, with an concommitant lack of interoperability, only one:

a) The frames for each traffic class group are identified by a distinct SCI, just as if they were transmitted by a separate port with its own SC.

Each SCI comprises 64 bits, the most-significant of which are expected to comprise a globally unique 48-bit MAC address, while the remaining bits are locally assigned. Existing reasons for assigning values to those bits include supporting virtual ports with a real port that has a single address, and allowing MACsec to operate, simultaneously, at more than one level[28] within an interface stack for a single real port. In the absence of new constraints on the least significant bits:

b) From the point of view of receiving SecYs, each of the traffic class grouping SCs is indistinguishable from SCs used by separate SecYs/ports to transmit.

c) Each data frame transmitted by a real or virtual port is only transmitted once, on a single SC, even if there are multiple traffic class grouping SCs.

So higher layer protocols, whose frames are protected by MACsec and whose purpose is to advertize the presence of systems or facilitate network topology computation, will not erroneously advertize more systems than actually exist[29].

The currently standardized MACsec entities can represent the above in two different ways. In one each transmitting traffic class grouping SC is supported by a separate SecY, just as if it was a separate member of a CA. However we do have to ensure that a given port only receives a given frame using one of its SecYs. In the other a real port has only one SecY, but that is equipped with as many traffic class grouping transmit SCs as it needs. In practice it is convenient to mix the two views by focusing on the individual aspects of SecY operation rather than using existing SecYs as our building blocks, the overall goal being to maximise interoperability with existing implementations and to reuse the current specifications including MIBs. The SCI[30], or (in the case of XPN[31]) an SSCI that is unique for each SCI, forms part of the IV used by each MACsec Cipher Suite, so:

d) Each traffic class group transmitting SC can use the same SAK (and sequence of SAs as traffic rolls over from one SA to its successor) just as members of a group CA do.

This leads directly to the second major point of the model. The existing, fundamental, notion of a CA—a Secure Connectivity Association comprising those ports that are in direct secure communication with each other—can be used without qualification. In particular:

e) The two (or more) traffic class group SCs associated with a single real or virtual port, together with the receive SCIs for that port, participate in the same CA.

This has the following consequences for key agreement and authentication:

f) The traffic class group SCs participate in the same MKA instance[32], identified by a single CAK, CKN tuple, and

g) the required CAK, CKN tuple is acquired using the same authentication exchange or provisioning step(s) that would normally be used for a port that used a single transmit SC. In particular there is no need to modify the way that EAPOL and EAP exchanges (in which the two real ports might participate as Supplicant and Authenticator) work, or to use additional exchanges[33].

[28] Protecting communication to a provider edge with a lower MACsec instance, for example, while protecting traffic to be carried over a service offered by the provider with another that ensures confidentiality and integrity independent of trust in the provider.

[29] There is a warning here for anyone who thinks that using MACsec to determine neighbors directly is preferable to using LLDP supported by MACsec.

[30] MKA ensures that each participant will use a distinct SCI so there is no risk of IV reuse with a given SAK.

[31] XPN: Extended Packet Numbering,Cipher Suites, allowing up to 2^{64} packets to be transmitted using a given SAK, before a new SAK is required.

[32] When reauthentication occurs the successfully reautheticated members of a CA start using an new CAK. It is convenient to view the continuing and uninterrupted communication as CA members overlap the use of new SAs supported by SAKs distributed with the new CAK with prior SAs as occuring within a single CA, rather than an overlap and replacement of CAs, particularly as the SCIs remain unchanged. For an up to date description of CAK and SAK rollover see P802.1Xbx.

[33] There is no need to distribute an explicit group CAK in the point-to-point case. The setings of the management variables joinGroup. formGroup, and new Group only relate to the distribution and acceptance of explicit group keys and have no impact on the use of a CAK derived from a point-to-point authentication exchange.

Each MKA participant, as currently standardized, represents a CA member that uses a single SC with its associated parameters such as the SCI and supporting SAs. In the interests of interoperability with existing implementations and of not creating new test cases:

h) Each traffic class group SC is represented by a separate MKA participant.

This approach complements the observation in (b) above. The bandwidth saving that would result from representing both SCs by a single MKA actor, transmitting a single MKPDU is negligible, and the upper bound on the number of MKA participants resulting from the need to represent each in an MKPDU live or potential peers list is not a problem for real-world deployment. Representing separate SCs by separate actors makes it easy for a running system to add and delete traffic class SCs.

3.2 MKA operation and implementation

It is quite possible, of course, to instantiate separate MKA actors for each of a port's SCs, copying received MKPDUs to both of them, executing their state machines separately, and ensuring that each receives the MKPDUs transmitted by the other. It is also possible to mimic two or more externally visible actors by extending the internal operation of a single actor implementation, as in the following description. This has advantages including:

— not complicating transmission and reception, as described above, and the associated scheduling;

— preserving the existing operation of the Logon Process[34] and its interaction with the CP state machine and the KaY

— using a single instance of the CP state machine thus avoiding any need to make decisions about distributing the responsibility for managing receive SAs, avoiding introducing complications into SAK and CAK rollover resulting from dependencies between the actors[35],

Each actor uses a separate MI and its own MN. The conditions under which either chooses a new MI remain unchanged, though each new MI should also be checked (and reselected if necessary) against those of its co-resident actors (representing the SCIs for other traffic classes).

The KaY maintains just one Live Peers List, and one Potential Peers List, using received MKPDUs to add to each in the usual way. When an actor transmits an MKPDU, its co-resident actors (representing the SCIs for other traffic classes) are added to the transmitted Live Peer List.

Only one actor, that associated with the default traffic class SC, should advertise itself as a potential Key Server, with the others advertising a Key Server Priority of 0xFF[36]. Similarly only the default traffic class actor should distribute keys in MKPDUs (if selected).

In the CP state machine RECEIVE state createSAs(lki) refers to the SAs used to receive from other members of the CA and all the transmit SAs used by the co-resident actors. Naturally there is no need to instantiate receive SAs for the latter. The 'electedSelf ...' transition from CP:RECEIVING is taken if satisfied by default traffic class actor[37]. In CP:TRANSMIT all the traffic class group transmit SAs are enabled.

The retireWhen timer in CP:TRANSMITTING takes care of the fact that a preempted frame may complete its arrival after a premepting frame has been received on a new SA from its transmitting port. This is legitimate when strict replay protection is not enforced.

NOTE—When strict replay protection has been configured there is, currently, a small window in which out of order (though not replayed) frames could be received as an SC rolls over from using one SA to its successor. This could be 'fixed' by adding a circular comparison on the AN number into the PN check, but implementing such a fix for a case that was not strictly advertized as part of the specification and is unlikely to occur with XPN CipherSuites (where SAs are likely to be used for the lifetime of a CAK, i.e. until reauthentication) is of dubious value. The obvious 'fix', discarding frames from an old SA after one from its successor has passed validation, is also not easy to apply to non-zero replay windows where there is some tolerance for misordering.

3.3 SecY and KaY Management

Both IEEE 802.1AE and 802.1X summarize their management objects in a UML diagram, provide a prose specification that matches and expands upon the UML, and specify SNMP MIBS that are intended to match the UML and prose. This section therefore considers the necessary changes to both the

[34] 802.1X-2010 Clause 12, 12.5.

[35] If the actors are implemented and scheduled separately, and one of them is a potential key server, the best results will generally be obtained by scheduling that actor first on MKPDU reception.

[36] See 802.1X-2010 9.5. The default traffic class SC should be retained even if SCIs for other traffic class groups are added to or deleted from a running system, so this decision minimizes the potential disruption as well as avoiding the need for a management variable for the other SCs. P802.1Xbx D1.6 clarifies a number of other issues related to Key Server selection that are equally applicable to the single and multiple actor cases.

[37] None of the other co-resident actors can be elected Key Server in preference.

UML/prose and the 802.1X PAE and 802.1AE SecY MIBs[38].

In 802.1X we are dealing with a small part of the management of the PAE, specifically that represented by the KaY class in the UML and the PAE KaY Group in the MIB. Each instance of the KaY class is contained within a PAE, multiple instances of which can be contained within a PAE System, each indexed by a portNumber. In the MIB a KaY is represented by an Ieee8021XKayMkaEntry in the ieee8021XKayMkaTable, indexed by port number. Each instance of the KaY has a single actor SCI, but otherwise aligns nicely with the implementation optimization suggested above (3.2). In particular the management view does not contain any of the ephemera (MI, MN, ...[39]) that might differ for individual co-resident actors responsible for different traffic class groups.

The SecY management text and UML provide a management view of receive and transmit SCs, specifying only one SC/SCI for transmit. Similarly the SecY MIB provides

— a secyRxSCTable that is a sequence of secyRxSCEntry's indexed by secyInterfaceIndex, secyRxSCI

— a secyRxSATable that is a sequence of secyRxSAEntry's indexed by secyInterfaceIndex, secyRxSCI, and secyRxSA. The last of these is the association number (AN) for the SA.

— a secyTxSCTable that is a sequence of secyTxSCEntry's indexed by secyInterfaceIndex

— a secyTxSATable that is a sequence of secyTxSAEntry's indexed by secyInterfaceIndex, and secyTxSA. The last of these is the association number (AN) for the SA.

NOTE 1—We need to enlist the help of a MIB expert when revising or adding to the MIB. The secyRxSCTable, secyRxSCEntry, and secyRxSCI are all MAX-ACCESS not-accessible. While compatible with using secyRxSCI as an index this seemed to omit vital functionality—use of the MIB to discover the values of the receive SCIs in the first place. I understand that SNP provides a method getNotAccessibleIndex()[40], and hope it works when a sequence of not accessible entries is in a not accessible table.

NOTE 2—The secyRxSAEntry's do not agree with the UML and the rest of the specification in one minor but possibly important respect. Each entry contains a secyRxSCcurrentSA, a pointer into the secyRxSATable. This might be taken as suggesting that only one receive SA can be in use (per receive SC) at a time, which clearly contradicts other parts of the specification. However the individual secyRxSAEntry's do contain an inUse indicator, matching the UML, so it is possible to see the overlap of the SA's. So the currentSA pointer is strictly unnecessary. It could also be argued that the time of overlap (bounded at 0.5 second by 802.1AE-2006 clause 7.1.3) is so short that it is not worth returning in response to a management query and that the pointer is a useful optimization.

First to be clear on the new functionality to be managed. The SecY should be capable of mapping the user priority (i.e. the priority associated with each Controlled Port ISS request) to:

a) the transmit SC (identified by its SCI) to be used for frames of that user priority, and

b) the access priority (i.e. the priority associated with the corresponding Common Port ISS request).

This is more flexibility than would be provided by mapping each of the Controlled Port user's traffic class queues to a transmit SCI and associating an access priority with that transmit SCI. While such a constrained mapping addresses some obvious uses of multiple SCIs with strict replay protection, it does not necessarily cover all reasonable uses. Differing priorities in a single traffic class queue may merit different handling and access to underlying services. Strict replay protection is not guaranteed to be the administrator's focus. Equally it is possible that, in the future, some application will require a more complex choice between one of a number of transmit SCs[41], so assumptions about priorities or traffic class groups should not be embedded in the MIB's TxSCEntry's (or the corresponding TransmitSC class in the UML).

The existing MIB's TxSC and TxSA tables and entries might be retained unchanged (or deprecated) and refer to the 'default' transmit SC, i.e. the one to be used if no others have been created by management and (by convention) the one to be used to support best efforts traffic. In that case an additional set of tables and entries might be created, as described below. If the addition of a further index to the existing object is permissible, with a default applying if the index is not

[38] MIBs do not always follow their base specifications, they have always been the domain of specialists and other have found checking them to be a tedious taskat best, largely as a result of the unremitting verbosity of the result. In general MIBs have deviated from the specifications they have meant to formalize for a number of reasons. MIB designs tend to follow certain grooves, the original specification may have been unclear to the MIB author who is unlikely to have participated in its development or be an expert in that particular subject, and the MIB authors may have received advice from others who simply think the specification should have said something else and who welcomed the opportunity to rework it. A future move to YANG might make it easier to identify and correct discrepancies. In the present case we have been fortunate.

[39] Other possible items?

[40] Access to objects that are 'not-accessible'. Obvious, really.

[41] I am not enthusiastic about increasing the number of SCs. When it comes to 'diffserv' use in general our 1995 prediction that eight traffic classes would be more than enough, and that two would meet most needs has been validated. Moreover it seems likely that most MACsec hops will continue to be order preserving and even on non-ordering preserving links it is not a given that network administrators will universally favour strict replay protection over the simplicty of using a single transmit SC. On the other hand we are seeing more activity in time-sensitive networking so perhaps, 25 years on, the 'intserv' model will see significant deployment-in which case we would not want to tie transmit SCs to firmly to traffic classes.

present, then we could make do with a single set of tables and entries. Presuming the first of these for the moment, the changes to the SecY UML would be as follows:

— Remove const SCI sci from the top-level SecY class. This is already superfluous since each SecY is identified (within the system as a whole) by its ControlledPort ifIndex/portNumber, not by its SCI[42].

— Change the link to the TransmitSC class from '1 transmitChannel' to '* transmitChannels'.

— Add a new transmitPriority class, with eight instances as children of the Generation class, each with const priority userPriority, priority accessPriority, and a link SCI transmitChannel to a TransmitSC.

That's it for SecY UML changes. The changes to the SecY MIB would be as follows:

— Deprecate the existing secyTxSCTable, secyTxSCEntry, secyTxSATable, and secyTxSCEntr, and add new tables and entries indexed in the same way as for the .

— Create new tables and entries for each of the above, indexing them exactly as for the receive SCs and SAs.

The 802.1X UML and MIB can remain unchanged, with the understanding that the KaY is to manage all the transmit SCIs required by the SecY and that the KaY's existing SCI parameter refers to the default SCI, i.e. the one configured for best effort traffic and the only one used if multiple transmit SCs are not required.

3.4 Interface stacks

The above would require no changes to interface stacks, either as they appear in the management model or as described in 802.1AE-2006 Figure 10-4. Minor editorial changes would be necessary in various places (e.g. Figure 10-3 Secure Frame Generation box should be explicit about transmit SC selection).

3.5 Other specification issues

802.1AE-2006 allows the SCI to be omitted from the transmitted SecTAG where the connectivity is point-to-point. This option has to be restricted to the point-to-point case where multiple SCIs are not in use[43]. Changes need to be made to allow operPointToPointMAC to be true even in the presence of multiple receive SCs (see 802.1AE-2006 10.7.4, 6.7, and A.7). IEEE 802.1AE-2006 preceded the development of MKA and its inclusion in 802.1X-2010, with the latter a better approach would be to have MKA decided whether the CA provides point-to-point connectivity or not.

4. Alternative solutions

4.1 No change

One approach is to leave the current specification unchanged, forcing the user of preemption to choose a suitably large replay window. If this has to be done on a case by case basis because a tight window is deemed desirable then the standardized management counters provide some help: if the management variable replayProtect is false then frames outside the replay window are counted as late but not discarded. This allows a network manager to get some sense of what the replay window should be, before discarding frames that are late.

While the Ethernet MAC itself might not limit the number of preempting frames transmitted while reception of a preempted frame is suspended, it is not plausible that one hundred per cent of the bandwidth has been allocated to high priority or time sensitive traffic for any other a short period. As soon as a preemptable but unpreempted frame is received the received stream will be at the leading edge of the window. The necessary replay window values will, therefore, be quite small.

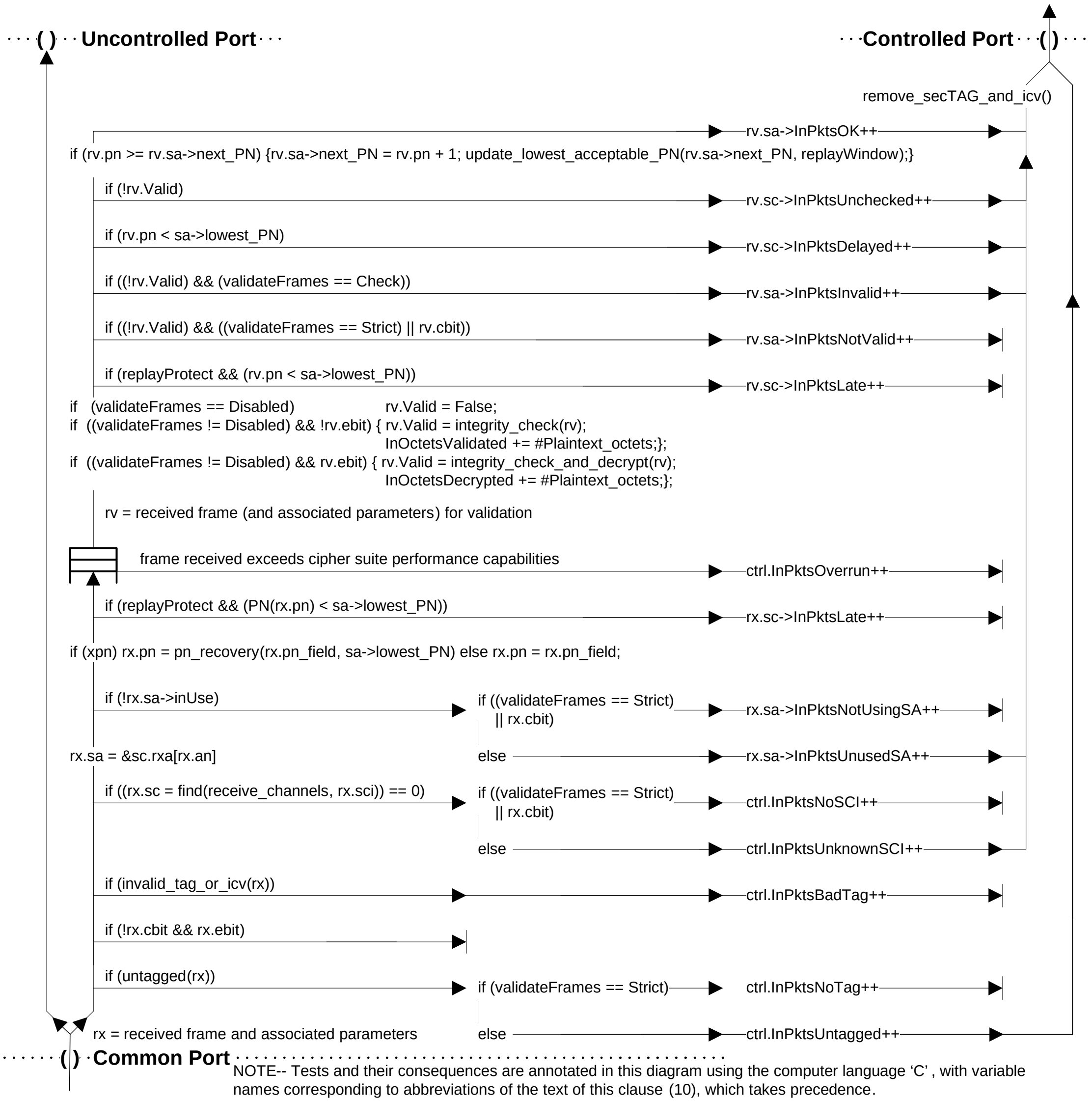
This 'no change' approach is made more attractive by the standard's procedures for bounding the time delay of transmitted frames—this reduces the intervals during which 'flapping' and related attacks might be carried out.

4.2 'Scoreboarding'

For implementation performance reasons it was decided during the development of 802.1AE-2006 that MACsec should rely on reducing the replay window to guard against replay and not also scoreboard, i.e. mark individual PNs within the window as received. Should this decision be revisited, taking into account implementation experience and technological progress, as an alternative to using multiple transmit

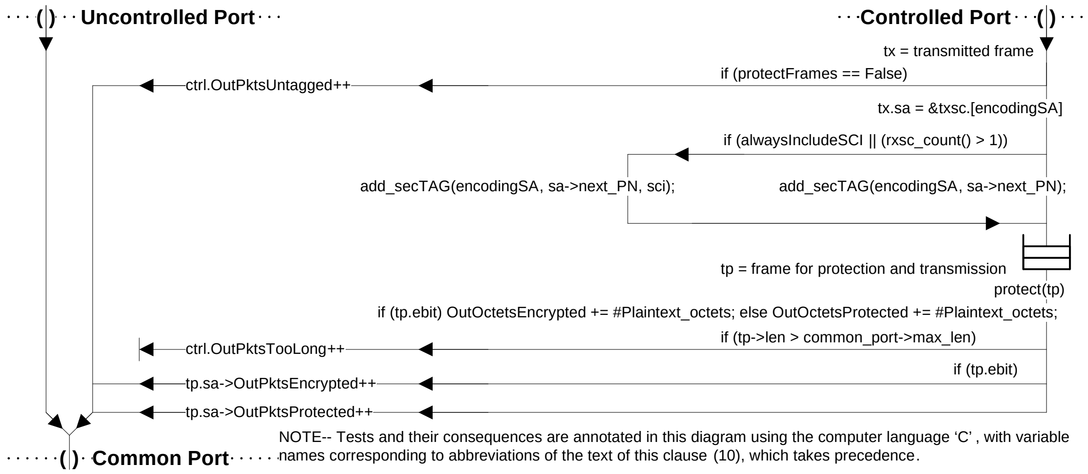
[42] There is a difference of approach between the UML management models in 802.1X-2010 (where the PAE System is the top-level class and port numbered PAEs are a sub-class of that) and in 802.1AE-2006

[43] This frame size saving option has not proved popular (acnecdotal evidence only, so far) so the option of having just the default transmit SC omit the SCI is not advocated.


( ) Uncontrolled Port
Controlled Port ( )
remove_secTAG_and_icv()
rv.sa->InPktsOK++
if (rv.pn >= rv.sa->next_PN) {rv.sa->next_PN = rv.pn + 1; update_lowest_acceptable_PN(rv.sa->next_PN, replayWindow);}
if (!rv.Valid)
rv.sc->InPktsUnchecked++
if (rv.pn < sa->lowest_PN)
rv.sc->InPktsDelayed++
if ((!rv.Valid) && (validateFrames == Check))
rv.sa->InPktsInvalid++
if ((!rv.Valid) && ((validateFrames == Strict) || rv.cbit))
rv.sa->InPktsNotValid++
if (replayProtect && (rv.pn < sa->lowest_PN))
rv.sc->InPktsLate++
if (validateFrames == Disabled) rv.Valid = False;
if ((validateFrames != Disabled) && !rv.ebit) { rv.Valid = integrity_check(rv);
InOctetsValidated += #Plaintext_octets;};
if ((validateFrames != Disabled) && rv.ebit) { rv.Valid = integrity_check_and_decrypt(rv);
InOctetsDecrypted += #Plaintext_octets;};
rv = received frame (and associated parameters) for validation
frame received exceeds cipher suite performance capabilities
ctrl.InPktsOverrun++
if (replayProtect && (PN(rx.pn) < sa->lowest_PN))
rx.sc->InPktsLate++
if (xpn) rx.pn = pn_recovery(rx.pn_field, sa->lowest_PN) else rx.pn = rx.pn_field;
if (!rx.sa->inUse)
if ((validateFrames == Strict)
|| rx.cbit)
rx.sa->InPktsNotUsingSA++
rx.sa = &sc.rxa[rx.an]
else
rx.sa->InPktsUnusedSA++
if ((rx.sc = find(receive_channels, rx.sci)) == 0)
if ((validateFrames == Strict)
|| rx.cbit)
ctrl.InPktsNoSCI++
else
ctrl.InPktsUnknownSCI++
if (invalid_tag_or_icv(rx))
ctrl.InPktsBadTag++
if (!rx.cbit && rx.ebit)
if (untagged(rx))
if (validateFrames == Strict)
ctrl.InPktsNoTag++
rx = received frame and associated parameters
else
ctrl.InPktsUntagged++
( ) Common Port

NOTE-- Tests and their consequences are annotated in this diagram using the computer language ‘C’ , with variable names corresponding to abbreviations of the text of this clause (10), which takes precedence.

Figure 1—MACsec receive processing (.1AE-2006 amended by .1AEbw-2013)


( ) Uncontrolled Port
Controlled Port ( )
tx = transmitted frame
if (protectFrames == False)
ctrl.OutPktsUntagged++
tx.sa = &txsc.[encodingSA]
if (alwaysIncludeSCI || (rxsc_count() > 1))
add_secTAG(encodingSA, sa->next_PN, sci);
add_secTAG(encodingSA, sa->next_PN);
tp = frame for protection and transmission
protect(tp)
if (tp.ebit) OutOctetsEncrypted += #Plaintext_octets; else OutOctetsProtected += #Plaintext_octets;
if (tp->len > common_port->max_len)
ctrl.OutPktsTooLong++
if (tp.ebit)
tp.sa->OutPktsEncrypted++
tp.sa->OutPktsProtected++
( ) Common Port

NOTE-- Tests and their consequences are annotated in this diagram using the computer language ‘C’ , with variable names corresponding to abbreviations of the text of this clause (10), which takes precedence.

Figure 2—MACsec transmit processing

SCs to retain order so a window of zero can continue to protect against replay? My personal point of view is that out-of-order frames are more likely to disrupt poorly designed or implementated applications than are simple duplicates, so it is not worth going back over this one.

4.3 Preliminary check only

In the anticipated preemption use case scenario, no further bridges are interposed between the MACsec transmitter and receiver and the initial fragments of each frame are received in PN order (see 2.9. above). Therefore the preliminary replay check, just before the fifo in Figure 1, can be used. While this will not enforce strict replay protection at all times, the receive fifo is bound to empty frequently since it is not possible to arrange for the applied load to match the service rate exactly for extended periods without risking overrun (4.3.). In terms of changes to .1AE all that is required is to remove or turn off the ‘if (replayProtect) && (rv.pn < sa->lowestPN))’ check that occurs after the receive fifo.

This was my preffered option while we were just talking about pre-emption, but does not address the larger service provider use case. The use of multiple transmit SCs deals with both, and makes it unnecessary to go into the fine modelling detail that might be required in the standard to discuss the receipt of frame fragments.