MACsec Counters

Mick Seaman

This note reviews the secure frame generation (transmission) and verification (reception) management counters specified by 802.1AE, and recommends reducing them in the light of current requirements. In particular, the introduction of traffic class SCs in P802.1AEcg will multiply the number of per SC and per SA counters, altering the balance between the implementation costs of counters and other storage requirements (e.g. for keys and key expansion tables, which SCs in the same CA share). The current counter set was designed to assist MACsec deployment. This included diagnosing misconfigured or untimely manual key distribution and misbehaving keying protocols, before the MACsec Key Agreement protocol (MKA) was standardized, and did not take into account either the information available from the latter or the way it uses MACsec.

1. Counter functionality

MACsec's frame generation and verification counters and configuration controls help to:

a) Assess the load placed on the cryptographic functions, and whether received frames are being discarded because the load is excessive.

b) Ensure that key management is working and being used correctly.

c) Ensure that the potential CA has the desired membership before possible participants are excluded.

d) Ensure that the replayWindow size is not aggressively small.

e) Detect discards due to unexpected delays or misordering.

f) Detect the presence of unauthenticated or unauthorized transmitting stations.

There is a strong focus on helping an administrator avoid network failure when first deploying MACsec in an existing network, with the consequent likelihood of security being turned off and subsequent deployment attempts restricted. Frames that would be discarded (for various reasons) by strict reception rules, are counted so the underlying issues can be addressed before those rules are applied. There are networks which should not operate at all if they are not secure, and for which a soft deployment approach is not required, but the current judgment is that it is not worth creating an option for that subset.

Note that in some (most?) cases a counter is used to monitor a present or potential condition that might equally have been tracked by a resettable flag or at least by a much smaller resettable counter. This is a quite normal use of counters, following the dictum "if it moves count it[1], and if it doesn't move count it to make sure it hasn't moved". Discussion as to whether such use is a good idea or not, with its accompanying arguments about multiple managers and the reliability of resets, is beyond the scope of this note. The latter confines itself to discussing whether specific counts could be omitted or physical counter resources used for different counters at different times, and does not suggest replacing counts with different tracking mechanisms.

2. Counter reduction opportunities

The opportunities for reducing the number of MACsec counters and their implementation costs include:

a) Combining currently separate counters where the current level of detail is no longer required. MKA provides better key management diagnostic information (as mentioned above), and it is now unlikely that a peer implementation of the MACsec tagging and cryptographic functions is simply incorrect.

b) Counting per SC, rather per SA. Roll-over from one SA to another is fairly rapid (at least on management timescales) and in any case MKA provides detail on the roll-over (see above).

Amongst these opportunities are those where an implementer with detailed knowledge, both of MACsec and of MKA, could have avoided dedicating a counters to each of the four possible SAs for an SC, dynamically allocating counters to the (at most) two in current use. When implementation responsibilities are divided between individuals, or indeed between organizations, the likelihood of such global optimization is diminished unless explicitly supported by the standard, which may be part of an

[1] That is to say count the move.

implementation agreement. An implementer hoping to partner with a number of others is unlikely to rely on his unsupported powers of persuasion to sell an optimized implementation.

There are also two per SA counters (which could be reduced to one, see below) that count frames received on the SA when it is not in use. This could be replaced with a single per SC counter, which might also be shared with other reasons for frame discard.

c) Not implementing physically separate counters where management and MKA controls dictate that only one could be incremented on a frame by frame basis.

From the point of view of management reporting it makes sense to give the counters different names, clearly identifying the meaning of each. However the cases where only one real-time counter needs to be kept should be clearly identified in the specification.

Early in the development of .1AE we attempted to stick to the principle of only counting each frame once. Unfortunately when soft deployment controls and counters are in use there can be a number of paths to frame reception. If aggregated reception counts were to support the RFC2863 MIB Interfaces Group separation of unicasts, multicasts, and broadcasts directly, then the counter for each path would become three counters—which is simply excessive. The MIB Interfaces Group counts of ifInUcastPkts, ifInMulticastPkts, and ifInBroadcast have to be maintained for the Controlled Port in addition to those specified for the secure frame verification (see .1AE Figure 10-5). This does mean that one (possibly more) of the latter is simply the sum of the IfIn counts and does not have to be accumulated in real time.

3. Counter reduction constraints

A far-reaching reassessment of counter requirements might result in one or more new counts that are not simply the sum of pre-existing counter values, effectively obsoleting prior implementations. In the case of MACsec these might be difficult to change, either because of their use of specialized hardware or because of a consequent need for recertification. This note avoids recommending any such changes.

4. Verification (reception) frame counters

Table 1 summarizes the existing frame counters for secure frame verification (i.e. reception), a proposed revised set, and variant of the latter. Table 2 gives the total number of counters used in a number of scenarios and implementation styles, with the number attributable to each Controlled Port as a whole and to each SC in each scenario shown below.

The description of the various conditions counted are taken from .1AE Figure 10-5, with a couple of simplifications (that are readily deducible from the table). The reduction in counters can be summarized as follows. First avoid counting per SA, rather count per SC or for the Controlled Port as a whole. Second, don't provide two or more counters when only one of them will be incremented for a particular setting of the management variable validateFrames (Disabled, Check, or Strict, with .1AEcg adding the–uninteresting from the point of view of the present discussion–value Null).

Each of the counters is subscripted R, D, or E. E identifies the counters which sum (for all SCs, SAs, and the Controlled Port as a whole–identified as CP) to the IETF RFC2863 MIB Counter ifInErrors (see .1AE 10.7.6). Similarly D identifies the counters that sum to ifInDiscards. R identifies counts of frames that are received Their sum is not an RFC2863 count, as that MIB distinguishes ifInUcastPkts, ifInMulticastPkts, and ifInBroadcast—an uninteresting distinction from the MACsec point of view—so three additional counters per Controlled Port are required to support the MIB.

Several of the cells in the 'Counter' columns of Table 1 contain two or more counters. This is an indication that only one of those counters is required at a time. The counter totals for the best current implementation (and for the proposed and alternative schemes) in Table 2 reflect that shared use. Strictly speaking the group of conditions currently counted by {InPktsNoSCI, InPktsUnknownSCI, InPktsNotUsingSA, InPktsUnusedSA} are not mutually exclusive if validateFrames is not Strict and a transmitter makes confidentiality or integrity-only per packet decisions. However that initial deployment corner case is hardly worth adding a counter for—the addition would have little diagnostic value. A single counter can be used for both the InPktsNotUsingSA and the InPktsUnusedSA conditions, and reported as the former (contributing to ifInErrors) if validateFrames is Strict or MKA has specified confidentiality with the last distributed SAK, and the latter otherwise.

Condition	Existing Counters		Proposed Counters		Alternative Counters	
	Counter	Per	Counter	Per	Counter	Per
untagged(rx) && (validateFrames == Strict)	ctrl.InPktsUntagged$_R$	CP	ctrl.InPktsUntagged$_R$	CP	ctrl.InPktsUntagged$_R$	CP
untagged(rx) && (validateFrames != Strict)	ctrl.InPktsNoTag$_D$		ctrl.InPktsNoTag$_D$		ctrl.InPktsNoTag$_D$	
!rx.cbit && rx.ebit	—	—	—	—	—	—
invalid_tag_or_icv(rx)	ctrl.InPktsBadTag$_E$	CP	ctrl.InPktsBadTag$_E$	CP	ctrl.InPktsBadTag$_E$	CP
unknown_sc(rx) && ((validateFrames == Strict) || rx.cbit)	ctrl.InPktsNoSCI$_E$	CP	ctrl.InPktsNotUsingSA$_E$	CP[1]	ctrl.InPktsNotUsingSA$_E$	CP[1]
unknown_sc(rx) && ((validateFrames != Strict) && !rx.cbit)	ctrl.InPktsUnknownSCI$_R$		ctrl.InPktsUnusedSA$_R$		ctrl.InPktsUnusedSA$_R$	
!rx.sa->inUse && ((validateFrames == Strict || rx.cbit)	ctrl.InPktsNotUsingSA$_E$	SA	ctrl.InPktsNotUsingSA$_E$		ctrl.InPktsNotUsingSA$_E$	
!rx.sa->inUse && ((validateFrames != Strict && !rx.cbit)	ctrl.InPktsUnusedSA$_R$		ctrl.InPktsUnusedSA$_R$		ctrl.InPktsUnusedSA$_R$	
replayProtect && (rx.pn < sa->lowest_PN || rv.pn < sa->lowest_PN)	rx.sc->InPktsLate$_D$	SC	rx.sc->InPktsLate$_D$	SC	rx.sc->InPktsLate$_D$	SC
!replayProtect && rv.pn < sa->lowest_PN && (rv.Valid || (!rv.cbit && (validateFrames == Disabled))	rv.sc->InPktsDelayed$_R$		rv.sc->InPktsDelayed$_R$		rv.sc->InPktsDelayed$_R$	
!rv.Valid && (validateFrames == Check)	rv.sa->InPktsInvalid$_R$	SA	rv.sc->InPktsInvalid$_R$	SC	ctrl.InPktsInvalid$_R$	CP
!rv.Valid && ((validateFrames == Strict) || rv.cbit)	rv.sa->InPktsNotValid$_E$		rv.sc->InPktsNotValid$_E$		ctrl.InPktsNotValid$_E$	
!rv.Valid && (validateFrames == Disabled) && !rv.cbit && !replayProtect && rv.pn >= sa->lowest_PN	rv.sc->InPktsUnchecked$_R$	SC	rv.sc->InPktsUnchecked$_R$	SC	rv.sc->InPktsUnchecked$_R$	SC
rv.Valid && (!replayProtect || rv.pn >= sa->lowest_PN)	rv.sa->InPktsOK$_R$	SA	rv.sc->InPktsOK$_R$		rv.sc->InPktsOK$_R$	

Table 1—MACsec verification frame counters

[1]Could keep 2 counters for this set, distinguishing received and discarded packets to allow for cases when confidentiality or integrity is applied on a per packet basis.

The proposed ‘not using’ and ‘unused’ SA counts are per Controlled Port, not per SA, because (a) the SA and indeed the SC assignments naturally cannot be verified, and (b) the counts are still a sufficient indicator of the failure of a participant to implement key agreement protocol correctly or of the presence of an outsider who is putting a SecTAG on frames.

The InPktsLate and InPktsDelayed counts are per SC, because the viable replayWindow sizes can be expected to vary both by the location of each peer participant in a CA (particularly when the MACsec-capable port is in an EDE, where the paths to each of a number of peers may differ significantly) and by traffic class, but there is no reason to expect a difference between an SA and its successor for a given SC. Similarly an attacker, or some accidental nuisance, might be present on one path and not on another, so InPktsOK, InPktsInvalid, and InPktsNotValid, are recorded per SC, which also makes it easier to correlate transmission and reception counts in a multi-participant SC.

Since inPktsUnchecked can share the same counter as InPktsOK it is not an implementation burden to make that a per SC count as well.

Changing InPktsInvalid and InPktsNotValid to per SC counts rather than per SA counts reflects the diminished probability that an MACsec implementation or the associated key management is both broken to the extent that a legitimate participant

Scenario	Counter Scheme and Implementation				
	Existing			Proposed	Alternative
	A[1]	C[2]	F[3]		
Point-to-point, single rcv SC	12	18	37	6	6
Point-to-point, two traffic class SCs	21	32	69	9	8
Point-to-point, three traffic class SCs	28	46	101	12	10
Multi-point (virtual shared media), 6 participants, no traffic class SCs	48	74	165	18	14
Multi-point (virtual shared media), 6 participants, two traffic class SCs per participant	93	144	325	33	24
Multi-point (virtual shared media), 6 participants, three traffic class SCs per participant	138	214	485	48	34
Per Controlled Port	3	4	5	3	4
+ per receive SC	9	14	32	3	2

Table 2—MACsec verification (receive) frame counters (per Controlled Port)[4]

[1]A denotes an intelligent implementation, counters per SC or SA as per standard, and the latter only kept for the two possible active SAs (except for noting packets on unused SAs).
[2]C denotes an average implementation, counters are per SC or per SA as called for by the standard (though counters are kept for all four possible SAs), a single counter is used when only one of a set could be incremented (as chosen by the management control validateFrames).
[3]F denotes a failing grade implementations, per SA counters are kept for each of the four possible SAs for each receive SC, even when the standard only calls for a per SC count. Separate counters are used even when only one of a set could be incremented.
[4]Totals exclude separate ifInUcastPkts, ifInMulticastPkts, and ifInBroadcastPkts for the RFC 2863 Interfaces Group MIB. These would be just for the/each Controlled Port, not per SC. Add 3 to each scenario cell to count these.

is using the wrong SAK for an SA, and the increased visibility (through MKA) of what SAs should be in use at any time. Counting per SC rather than per SA does not reduce the ability to detect and localize attacks and nuisances.

5. Generation (transmit) frame counters

There are rather fewer transmission counters (see Table 3 and .1AE Figure 10-4) and their implementation can also be optimized (see Table 4).

When an SA is created (see .1AE 10.7.21 and Figure 10-6) confidentiality or integrity is selected, so all frames transmitted for that SA will be either encrypted or subject to integrity-only protection (if the management control validateFrames is not False). The frames protected for a given SA will therefore contribute to the OutPktsEncrypted or the OutPktsProtected count, but not both. The number of packets that have been transmitted on an active SA is already available to management (.1AE 10.7.14) through the value of nextPN[2] (the packet number PN is part of the IV used by the Ciphersuite for the SA).

In theory nextPN derived counts for encrypted and protected packets should be adjusted to allow for frames that are discarded as too long[3] (after the addition of the SecTAG and ICV) for the underlying Common Port's MAC Service, and counted in OutPktsTooLong, but there is little utility in recording the difference. In particular the discard of over long frames will almost invariably be followed by some remedial packet size adjustment by the original transmitter. The standard should explicitly permit (but not require) the use of nextPN without adjustment. With this accommodation there is no need for the secure frame generation process to implement additional counters that need to be rapidly updated apart from the per Controlled Port OutPktsTooLong. If validateFrames is False the total number of frames transmitted is to be reported, however if the Interfaces Group MIB (RFC 2863) counts are to be kept for Controlled Port this is the sum of the unicast,

[2]The very small time delay between nextPN being incremented and a frame being transmitted is of no interest on management timescales.
[3]The entire frame is not necessarily available when cryptographic processing begins, so the PN for that frame (and for following frames in a pipe-lined implementation) can be assigned before it is known that the frame will have to be discarded.

multicast, and broadcast MIB counts. If the MACsec implementation is simply providing a single Controlled Port over a single real physical port, these can be derived from those for the Common Port implementation, adjusted for those packets transmitted (probably relatively few) by the Uncontrolled Port (which probably has but a single client, the PAE—supporting authentication, authorization, and key agreement). However if the MACsec implementation is supporting a number of virtual ports, as in the multi-access LAN scenario (.1AE 11.8), further Controlled Ports have been brought into being and the set of three transmission counters (unicast, multicast, and broadcast) needs to be added for each.

Condition	Existing Counters		Proposed Counters (no change)		Alternative Counters (place-holder)	
	Counter	Per	Counter	Per	Counter	Per
protectFrames == False	$ctrl.OutPktsUntagged_T$	CP	$ctrl.OutPktsUntagged_T$	CP		
(protectFrames == True) &&	$ctrl.OutPktsTooLong_D$	CP	$ctrl.OutPktsTooLong_D$	CP		
(protectFrames == True) && tp.ebit	$tp.sa.OutPktsEncrypted_T$	SA	$tp.sa.OutPktsEncrypted_T$	SA		
(protectFrames == True) && !tp.ebit	$tp.sa.OutPktsProtected_T$		$tp.sa.OutPktsProtected_T$			

Table 3—MACsec generation (transmit) frame counters

Scenario	Counter Scheme and Implementation			
	Existing		Proposed	Alternative (place-holder)
	A[1]	D[2]		
Point-to-point, single rcv SC	1	10	1	
Point-to-point, two traffic class SCs	1	18	1	
Point-to-point, three traffic class SCs	1	26	1	
Multi-point (virtual shared media), 6 participants, no traffic class SCs	1	10	1	
Multi-point (virtual shared media), 6 participants, two traffic class SCs per participant	1	18	1	
Multi-point (virtual shared media), 6 participants, three traffic class SCs per participant	1	26	1	
Per Controlled Port	1	2	1	
+ per transmit SC	0[3]	8	0[3]	

Table 4—MACsec generation (transmit) frame counters (per Controlled Port)[4]

[1]A denotes an intelligent implementation, already using nextPN.
[2]D denotes a poor implementation, separate counters are used, do not take advantage of the fact that ll frames for a single SA will receive the same protection (confidentiality or integrity-only), and the fact that OutPktsUntagged can be derived from the Interfaces MIB count is ignored.
[3]Counting only real-time management counts, i.e. ignoring nextPN which is part of the basic MACsec generation/protection, and ignoring roll up counts that only need to be done on a timescale of minutes or on demand.
[4]Totals exclude separate ifInUcastPkts, ifInMulticastPkts, and ifInBroadcastPkts for the RFC 2863 Interfaces Group MIB. These would be just for the/each Controlled Port, not per SC. Add 3 to each scenario cell to count these.

Reporting the nextPN derived OutPktsEncrypted and OutPktsProtected counts allows these transmit counts to remain SA based. This retains detailed diagnostic capability for SA rollover problems, supplementing information carried by MKA, though if counting per active SA was not essentially free an argument would

be made for counting on an SC basis. The per SA counts are, of course, rolled up into per SC counts. Comparing transmit and receive SC counts provides information into locality, path, or traffic class dependent issues.

6. Cryptographic performance counters

The verification and generation counters discussion above omits those that monitor cryptographic performance. The InPktsOverrun count (.1AE Fig 10-5) serves two purposes. First it explains and quantifies frame loss for a particular Controlled Port client. Second, taken together with the other cryptographic counts, it provides information on cryptographic performance limits—when they have been reached. However in many implementations this counter will remain at zero, and there is no point or need to implement a dedicated count. It may be that the MACsec implementation is capable at operating at full wire speed. Alternatively it may be that the effect of less than wire speed processing leads eventual to an input queue overflow earlier in the received pipeline, at a point where that is indistinguishable from the effect of the eventual client failing to process received packets fast enough.

Except in the unusual case when they are many late or invalid packets, the octet counts InOctetsValidated and InOctetsDecrypted will roughly correspond to the number of octets delivered to the Controlled Port, with only one of those counts being incremented for a given SA, and only one incremented for all SAs using the same SAK when MKA is used. How these counts are best collected is implementation dependent, and implementations that support many Controlled Ports with a single cryptographic resource should probably choose to provide statistics that give a better idea of how close to (or over) the performance edge they are operating.

7. Other issues

In reviewing Figure 10-4 I noticed that lowest_acceptable_PN counter is not updated if the test “if (!rv.Valid)” (near the top of the figure) succeeds. However the frame is still received in this case, since it only applies if the management control validateFrames is Disabled and the frame has not been confidentiality protected. The test should have been applied after the conditional update of lowest_acceptable_PN, in its present position there will be no useful information as to whether the replayWindow size is adequate until validateFrames is set to Check or Strict.

8. Conclusion and Recommendations

Add the following changes to 802.1AE to 802.AEcg:

1) Rename verification counts in Figure 10-5 as proposed in Table 1, adding explicit notes about which counts will be mutually exclusive and also allowing ‘NotUsingSA’ and ‘UnusedSA’ to use the same counter.
2) Move the check on updating lowest_acceptable_PN in Figure 10-5 (as noted in Section 7).
3) Add notes on the implementation of generation counters to Figure 10-5 and 10.5.
4) Update 10.6 and 10.7 to match the above, putting necessary normative statements in the text.
5) Update the MIB, and the MIB object cross-reference tables in Clause 13.
6) Update the PICS to match.
7) Create an informative annex that is a (much reduced) version of this note. Include explanation of how the new reduced set of counters can be supported by adding up counters provided by prior conformant implementations.