MKA optimization for group CAs

Mick Seaman

MACsec Key Agreement (MKA, Clauses 9 through 12 of IEEE 802.1X-2020) explicitly supports group connectivity. It provides a secure fully distributed multipoint-to-multipoint transport and applications of that transport including distribution of data keys (SAKs) by an elected Key Server. Each participant transmits and receives MKPDUs using a group address, thus communicating with all the others and reducing the number of MKPDUs required to add a new participant to an existing group.[1] Each of the participants can cryptographically validate MKPDUs transmitted by any of the others, supporting direct timely communication to support (for example) early identification of an alternate Key Server and (for another example) delay bounding of transmitted data. The Key Server distributes each SAK,[2] identifies the participants that are to use it and their readiness to receive from each of the others, and initiates protected data transmission. Participants can or could (with appropriate standardization) reduce the processing required to validate MKPDUs and install keys. This note describes some possibilities, and pitfalls.[3]

1. Overview

This note describes:

— How MKA participants in a potential or current group CA[4] might reduce the effort required to validate MKPDUs (2) and/or increase protocol responsiveness without increasing the effort expended to protect MKPDUs (3).

— The existing procedures for SAK distribution, the requirements and existing rules for fresh SAK distribution, and the need for good random number generation to support those requirements (4).

— The participant effort required for fresh SAK installation and use (4.5).

— How prior knowledge of the likely or required[5] CA participants and a common set of performance goals can expedite rapid group CA formation, and a possible Cipher Suite independent addition to the existing MKA TLVs that would allow SAK installation in parallel with other operations (5).

— Existing and possible Cipher Suite dependent ways of extending the nonce space to allow a new or changed set of participants to continue to use an existing SAK (6).

While some of the optimizations described do not change or add to the syntax and semantics of MKPDUs and TLVs, they should still be subject to the scrutiny and documentation that comes with standardization—verifying that they do address a real need not met by the existing standard or other optimizations, do not conflict with the latter, and do not depend on further changes. Any substantive changes should be broadly applicable, standardized as such, and not justified by special case or scenario pleading—not least because secure implementations need thorough validation[6], but also because scenarios can change in unexpected ways.[7]

[1] The group address used is generally one of the Reserved Addresses specified in IEEE Std 802.1Q, each of which has a defined scope i.e. frames with that destination address are filtered by certain bridges. This reduces the risk of an 'attack from a distance' and of accidentally creating unwanted obscured secure connections. A potential attacker needs the help of an insider (within the circumscribed scope) to interfere with MKA even if hat attacker possesses the CAK (possibly by prior equipment theft).

[2] SAK (Secure Association Key), a data key used by MACsec to protect frames.

[3] This note follows up on a brief discussion in the 802.1 Security Task Group, November 2024. It should be regarded as a living document, a work in progress. All statements in this note represent the personal opinion of the author, not that of the IEEE, or the 802.1 Working Group or Task Groups.

[4] Secure Connectivity Association.

[5] Required in the sense that their participation in network operation is necessary for successful networked application operation.

[6] Validation not just of the code of the implementation but also of any imported system libraries and calls, and of the entire tool change (compilers, linkers, etc. etc.).

[7] Any special pleading based on timing accuracy, including network speed, is particularly suspect. A 10 Mb/s network could well benefit from bridging to higher rate technologies in the future (which is one of the reasons why a single default Cipher Suite, capable of very high speed operation, was chosen for 802.1AE-2006). At such a future time it might be impractical or impossible to update original lower speed stations.

2. Selective MKPDU validation

MKA participants other than an elected or aspiring Key Server can omit, or treat as lower priority, validation of certain MKPDUs, as follows.

MKPDUs are not confidentiality protected. This was a deliberate design decision so that protocol operation, and any difficulty in the progress of that operation, could be observed by a network administrator who did not possess the CAK.[8] Each MKPDU's content can be inspected and used to decide whether it should be validated. MKPDUs can be retained for later validation (subject to ageing out) if required.

Selective validation is definitely safe if no protocol action is taken on an unvalidated MKPDU, as the current attack model assumes [see item a) of 9.1 of 802.1X-2020] that an attacker can selectively prevent the delivery of any frame. It could be used to reduce the validation workload of any participant, however there are some more or less obvious consequences worth spelling out (2.1–2.5).

2.1 Duplicate MI detection

Each participant needs to check the content of MKPDUs transmitted by other participants for duplicate use of its own MI to identify that participant as specified in 9.4.2 of 802.1X. Before taking any action as a consequence of apparent duplication, the MKPDU in question needs to be validated.

2.2 Maintaining liveness

MIs (Member Identifiers) can only be added to a participant's Potential Peers or Live Peers List as a consequence of receipt of an MKPDU that passes validation. A participant that does not respond to any Key Server MKPDU for 3 seconds,[9] risks being aged out and excluded from future communication. The last unvalidated Key Server MKPDU needs to be retained in case it needs to be validated so an MKPDU including its MI.MN can be sent.

2.3 Confirming connectivity

The presence of any other participant's MI in a transmitted MKPDU's Live Peer List is an indication that the transmitter has received a recent MKPDU from the participant that includes the transmitter's own MI and recent MN. That confirms a direct, live, data path from the other participant to the transmitter. If two participants, Alice and Bob (say), include each other's recent MI and MN in their respective transmitted MKPDUs then there is a potential (at least) direct secure data path between the two.

Optimizing out receive validation of MKPDUs from participants other than the current Key Server (or perhaps its potential successor) removes this indication of connectivity, and a network administrator needs to be aware of this potential reason for the absence of expected Live or Potential Peer List entries. For that reason, if not other, it might be advisable to standardize a selective validation optimization, if thought to be generally useful. While it could be used without any change to protocol fields, it might be wise to provide an indication of its use in MKPDUs.

If a participant has validated only MKPDUs recently transmitted by the Key Server (within MKA Life Time and MKA Life Time plus MKA Hello Time, see 9.4.3 of 802.1X), then the only peer on its Live Peers List will be the Key Server, and the only peers on its Potential Peers List will be those received on the Live Peers List of Key Server MKPDUs.

2.4 Total number of MKPDUs transmitted

Since MIs received in unvalidated MKPDUs cannot be added to a participant's Live or Potential Peer Lists, a failure to validate received MKPDUs from participants other than a Key Server (Kevin, say) can result in the transmission of more MKPDUs in total as participant Bob (say) cannot learn Key Server Kevin's MI from Alice's MKPDUs, but is reduced to exchanging MKPDUs directly with Kevin. Detailed analysis of this potential inefficiency depends on the number of participants, the intervals at which they can be expected join, and the effort required for MKPDU reception compared to that for validation. Kevin's transmission strategy also plays a part—a single MKPDU can be transmitted in response to initial transmissions from several other participants—so Kevin can improve upon an independent 3-way handshake with each of the other participants.

[8] The secure Connectivity Association Key (CAK) is either pre-shared/pre-place key (PSK) or a direct or indirect result of a prior authentication exchange, demonstrated live possession of which is the token of prior authentication and authorization. See 6.2 of 802.1X for a description of the key hierarchy. Allowing any network administrator to observe MKA operation without knowing the CAK significantly reduces the attack surface. Where CAKs are securely distributed, or calculated, as a result of an authentication exchange there should be no need to make the CAK, or the ICK and KEK, available outside of as secure partition within each participant that derives and uses the ICK and KEK to protect and verify MKPDUs and to wrap and unwrap SAKs (data keys). MKA does not reveal information that compromises privacy beyond that which an attacker can glean from observing protected data frames, even when MAC Privacy protection is used to reduce the information disclosed by data frames.

[9] MKA Life Time (Table 9-3 of 802.1X-2020) is 6 seconds. I suggest 3 seconds for response time to guard against potential MKPDU loss, but that figure could be refined. In stable operation the Key Server will transmit at MKA Hello Time (2 second) intervals, so simply discarding an MKPDU id a bet that the next periodic transmission will be received.

2.5 Peer determination

Unless extended packet numbering (XPN) is being used, the MACsec nonce comprises an SCI (Secure Channel Identifier, the transmitter's MAC Address followed by a port number) and a 32-bit packet number (PN). The SCI is either encoded in the SecTAG of each MACsec protected frame or derived on receipt from the frame's source MAC Address (9.3, 9.9, 14.1 of 802.1AE).

The SCI is used, together with a two-bit Association Number (AN) encoded in the SecTAG of each received MACsec-protected frame to associate the frame with a Secure Association (SA) and then to identify and update the lowest acceptable PN for the SA, discarding frames not within the replay window (potentially enforcing in order delivery). Frames not associated with a known SC/SA are discarded prior to MACsec validation (if validation is required, see validateFrames == Strict in Figure 10-4 of 802.1AE).[10]

The SCI of each peer is not included in MKPDUs transmitted by the Key Server. The mapping between each any given peer's MI (which is included in the Key Server's Live Peer List) and the corresponding peer's SCI is available in each of the MKPDUs transmitted by the peer. While there is no subsequent MACsec-protected frame data integrity or confidentiality exposure in taking the mapping from one of the latter MKPDUs without validating it — if it was sent by an attacker that did not in fact possess the SAK, any subsequent apparently MACsec data frames sent by that attacker will not pass validation — that would increase an attacker's DoS options and present a confusing management picture. On balance, and considering 2.4 above, it seems wise to continue to require validation of any MKPDU prior to taking any protocol action, including peer SCI determination.

NOTE—A received MACsec protected frame, sent by a CA participant possessing the SAK, could be validated without assigning it to an SA. So it would be possible create the SA, and to assign an initial lowest acceptable PN value purely on the basis of receiving the frame. The failure to follow the processing order specified in 10.6 of 802.1AE could be considered harmless. However it could also be impractical for hardware based MACsec implementations.

3. Repeated MKPDU transmission

Hand-in-hand with possibility of selective MKPDU validation is the possibility of repeated transmission of the same MKPDU, specifically by a Key Server that is attempting to facilitate rapid instantiation of secure connectivity between potential CA participants whose arrival is likely to be roughly but not exactly synchronized by power supply availability. If individual participants check the in-clear data of presumptive Key Server MKPDUs, and their cost of reception and such checking is acceptably low, the Key Server can repeat the same MKPDU, possibly without any Live or Potential Peer List entries, until some target time has elapsed or a satisfactory number of responses have been solicited. Only then might the Key Server update its Live Peer List, possibly distributing an SAK at the same time.

As with selective validation, repeated transmission is safe, from a security point of view, as the current attack model assumes [see item a) of 9.1 of 802.1X-2020] that an attacker can copy any frame and transmit arbitrary frames (except of course frames never previously transmitted and whose construction would require knowledge of the CAK derived keys).

Rapid repeated MKPDU transmission addresses the possibility that some participants transmit their initial MKPDUs after power up when they remain unable to receive from others who have already transmitted their initial MKPDUs. In an ideal world that doesn't happen, but there may be active intermediate components of the LAN infrastructure that power up after the attached stations.

4. SAK distribution detail

The discussion so far has suggested lowering the MKA workload for non-Key Server participants by reducing the effort they expend in MKPDU validation (2. above). That effort might be considered (by some) excessive in two general cases: (a) when a very large number of participants are involved;[11] and (b) when very rapid CA [12] formation is desired after some more or less synchronizing event, such as near but not exact power cycling of the attached participants causing the loss of prior {SAK, PN, MI, MN} state.

[10] Other settings of the management variable 'validateFrames' allow validation to be skipped, with or without SecTAG and ICV removal, or forwarding of invalid frames. These settings were more relevant prior to MKA standardization, anticipating potential issues with non-standard key agreement protocols and wishing to avoid mandating combined MACsec/MAC implementations which could prove unusable if those protocols failed.

[11] The current limit as to the possible number of participants is effectively determined by the inclusion of each of their Member Identifier.Member Number (MI.MN) tuples in one or other of the Live or Potential Peers Lists. At 16 octets per peer, that works out to a little less than 100 participants in a CA (secure Connectivity Association). If each transmits at MKA Hello Time (2.0 seconds, Table 9-3 of 802.1X) that implies a constant validation rate of about 50 MKPDUs/second. Note that I do not intend to imply that sharing SAKs amongst such a large group is a good idea.

[12] In this context CA stands for secure Connectivity Association, created by the use of MACsec over the insecure Connectivity Association created simply by attaching end stations to the same (possibly bridged) LAN media.

Rapid CA formation can also be expedited by frequent repeated MKPDU transmission by a Key Server (3. above), using selective validation to mitigate the potential workload increase.

4.1 Basic SAK distribution and rollover

Figure 1 and Figure 2 illustrate simple MKPDU exchanges for SAK distribution and installation[13] (notation and scenario adapted from 9.17 of 802.1X).

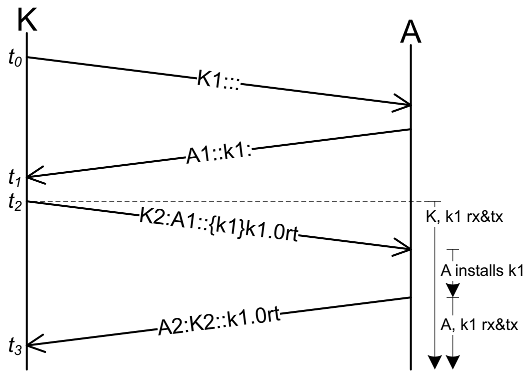

Figure 1—Initial SAK Distribution

Figure 1 begins with an MKPDU transmission, after power up, from Key Server, Kevin, to participant Alice. Since Kevin and Alice can complete power up at different times, it is likely that a prior MKPDU has been lost.[14] Kevin's first MKPDU, transmitted at t_0, contains just its MI and first MN (shown collectively as 'K+1') with the empty Live and Potential Peer Lists (shown after the first and second ':' respectively) and no additional data (after the third). Alice responds with its own MI.MN ('A+1'), an empty Live Peer List, an a Potential Peer List containing just K+1. Receipt of that message proves Alice's liveness to Kevin, who can then (at t_2) transmit an MKPDU including a key wrapped SAK (the Key Number is shown as 'K+1', but is not tied to Kevin's MI or any MN) and the MI.MN received from Alice.

Kevin includes the SAK Use parameter set in its MKPDU, with the Key Number, Association Number (AN, '0' in this case) and the transmit and receive status ('rt' , showing both enabled) for the Latest Key.[15] In this scenario, as described in 9.17 of 802.1X, there is no prior SAK and Kevin enables both receive and transmit immediately ('.rt') for the Latest (and only distributed) SAK. The text of 9.17 says that Kevin loses nothing by immediate use of the SAK without waiting for any acknowledgment that Alice can receive using the SAK, as there was no prior communication. However that statement does rather rely on the use of well designed and symmetric initial configuration protocols—so if Alice misses a first message from Kevin, Alice will transmit after installing the SAK and being able to receive subsequent messages from Kevin without waiting for Kevin to timeout.[16] The last MKPDU in the figure merely advertises Alice's transmit and receive status—its receipt is not a vital part of enabling MACsec-protected communication.

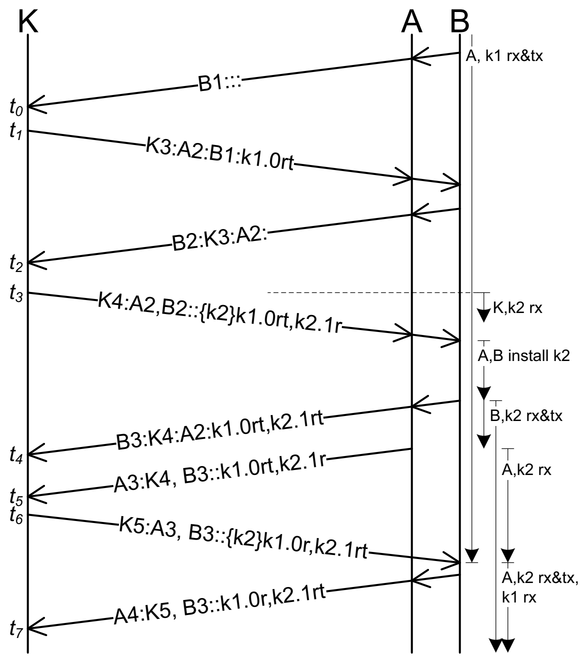

Figure 2—Follow up SAK Distribution

Figure 2 shows a continuation of the dialogue, with a third participant, Bob, joining, which forces (9.8 of 802.1X) the distribution of a fresh SAK (K+2) at t_3. Alice has to receive, and validate, two MKPDUs from Kevin—one with the fresh SAK, and one indicating

[13] Scenarios from 9.17 of 802.1X, notation abbreviated.

[14] A first MKPDU from A might also precede the sequence shown, effectively prompting K to begin (K would then include A's MI and MN in its initial Potential Peer List transmission).

[15] Key Numbers are, of course, included so there is no doubt to which SAK these status bits refer.

[16] Lossless initial startup and roll-over could benefit from refinement of the CP:READY to CP:TRANSMIT transition conditions taking into account the number of peers on the Key Server's Live Peer List when a SAK is distributed. If there is just one (i.e. the secure connectivity is to be point -to-point), and the Key Server is already able to receive when it sends the MKDPDU with the first (initial SAK), then the peer participant can starting transmitting immediately without loss, and the Key Server can begin transmitting when it receives an MKPDU from that peer (or any data). If the secure connectivity is multi-point (i.e. there are more than entries in the Live Peer List), lossless startup can be achieved by the Key Server waiting to transmit until it sees MKPDUs from all participants indicating that they can receive using the SAK, wile those participants wait to transmit until they receive an MKPDU indicating that the Key Server is transmitting, or any data frame from the Key Server or another participant using the new SAK. The same conditions apply to lossless rollover.

that Kevin has started transmitting using that SAK, so Alice can also proceed with transmission using that SAK (transition from CP:READY to CP:TRANSMIT in Figure 12-2 of 802.1X). After the first, Alice transmits an MKPDU when it has installed SAK K+2 for reception,[17] allowing Kevin to transmit the second, coordinating the lossless rollover from K+1 to K+2.

If the addition of participants to a CA is spread over time, the pattern of communication for existing participants on each addition will follow Alice's in Figure 2. Each receives a new SAK, installs it, enables reception, transmits an MKPDU, and after they have all enabled reception with the new receives the go ahead to transmit, and reports its status. The last of these need not be prompt, but can occur as part of periodic transmission. The MKPDUs transmitted by the Key Server are multicast, not per participant, so each participant addition results in just one MKPDU from each of the existing participants (reporting key installation).

The timing of fresh SAK distribution is restricted by item c) in 9.8 of 802.1X — a fresh SAK can be distributed if MKA Life Time (2.0 second) has elapsed since the prior SAK was first distributed, or if the Key Server's Potential Peer List is empty. If new participant arrivals occur at intervals that are shorter than the minimum between the Key Server's attempts to distribute SAKs, they will result in the distribution of a single fresh SAK after they have all be added to the Key Server's Live List. The Key Server cannot, of course, distribute fresh SAKs faster that it can install them itself. However there is no requirement in 9.8 for the Key Server to wait until all Live List participants have reported successful installation of a given SAK before distributing a fresh SAK as such a requirement would not cope with possible participant failure.

If distribution of a fresh SAK does address the arrival of several new participants, as in the immediately prior paragraph, then it might be distributed and brought into service with as few as two MKPDU transmissions per new participant, one from each of the existing participants, and two from the Key Server. The operative word here is 'might', as the first MKPDU from each new participant needs to include a recent MI.MN for K in its Potential Peer list. That could be obtained from an MKPDU with a non-null Live List transmitted by an existing participant, after validating that MKPDU

4.2 SAK distribution requirements

The principal requirement for the distribution of fresh SAKs stems from the absolute need to avoid the use of the same SAK and cryptographic nonce[18] to protect data frames that differ in any way. The nonce space for non-XPN Cipher Suites is subdivided by SCI, as described in 2.5 above. The nonce space for XPN Cipher Suites is subdivided by a shorter SSCI,[19] with successive MIs in the Key Server's Live Peer List allocating successive SSCIs to their participants.[20]

In addition to distributing fresh SAKs to new participants, and when any participant's packet number (PN or XPN) part of its nonce space nears exhaustion, the Key Server is responsible for ensuring that no two entries in its Live Peer List represent different participants with the same SCI[21] when an SAK is distributed. Even if the nonce space is partitioned by SSCI (rather than SSCI) for transmission/protection, the SSCI is indexed by SCI on reception, so a frame from Bob, say, could be discarded as duplicate of (or outside the replay window set by) a prior frame from Alice.

When a new participant joins the CA, a fresh SAK (or at least a fresh SAK.nonce space tuple) is also required to guard against replay of old frames to that new participant.

A further goal of fresh SAK distribution is to reduce the window for individual participant compromise, under the assumption that a participant that is no longer operational on the network might be the subject of some interference.

Finally of course, fresh SAKs are required when participants are reauthenticated, and the CAK and SAK rollover procedures in 802.1X-2020 (and 802.1X-2010) handle that possibility without MACsec-protected data frame loss.

4.3 SAK distribution rules

The requirements for fresh SAK distribution stated in 9.8 of 802.1X-2020 are deliberately cautious in two respects: mandating fresh SAK distribution on almost any change in the Key Server's Live Peer List, while at the same time restricting the rate of fresh SAK

[17] Bob also transmits an MKPDU to indicate successful installation of the new SAK (not shown in the Figure).

[18] IV, or "initialization vector" for GCM and GMAC.

[19] 'SSCI' literally 'Short Secure Channel Identifier'.

[20] With, for MKA version 3, the position of the Key Server SSCI also carried explicitly in the Live Peer List parameter set (Figure 11-9 of 802.1X-2020).

[21] Given the prevalence of network management operations that can change a station's MAC Address, the possibility of local MAC address assignment which might be by random selection, and the weakness of some vendor allocation procedures this is a real possibility, inadvertently or as a consequence of a planned attack.

distribution to provide ample time for each to be installed. The assumption in that current text is that a SAK has been ‘distributed’ when it is included in a Distributed SAK parameter set (Figures 11-11 and 11-12 of 802.1X-2020). What is important for preventing nonce reuse with a given SAK is the installation and use of the SAK for transmission. When a participant is rolling over protected data transmission from one SAK to its successor, transmission with that successor is delayed until the Key Server sets ‘Latest Key tx’ in its MACsec SAK Use parameter set. That delay allowed the Key Server to observe the ‘Latest Key rx’ for prior and new participants and avoid (for any participant that installs the new SAK promptly) protected data frame loss during rollover. However a new participant can use the new SAK immediately (see the transition from CP:READY to CP:TRANSMIT when !controlledPortEnabled in the CP state machine, Figure 12-2 of 802.1X-2020). The Cipher Suite independent rules in 9.8 of 802.1X-2020 thus prohibit subsequent redistribution of the SAK, i.e. the inclusion of that SAK in a Distributed SAK parameter with a different Live Peer List or a different Key Number.[22]

4.4 RNG considerations

Each MKA participant is required to use a fresh, randomly generated 96-bit MI whenever it starts or restarts. This is essential if does not have a record of the highest MN used or received with the current MI,MAC and so can no longer screen received MKPDUs including that MI to check that they have been transmitted by a currently live peer and include the freshest information distributed by that peer.[23]

MKA’s threat model [item a) in 9.1 of 802.1X] includes attackers that can selectively prevent delivery of frames to some participants, can copy frames (including MKPDUs), and can transmit arbitrary frames to arbitrary frames. An attacker could record MKPDU exchanges between a participant and a legitimate Key Server. So, if the participant restarts (as evidenced by its MN re-use) with the same MI, the attacker could replay the recorded Key Server MKPDUs in apparent response to those sent by the participant, inducing it to install a previously used SAK. If that SAK is then used by the participant with a previously used Cipher Suite nonce and a different data frame an attacker could use those frames to recover the SAK and gain receive and transmit access to previously secure communication. Given the aforementioned attack capabilities, other CA participants might be unaware of the intrusion.

So, a potential challenge for rapid CA formation, or extension, incorporating newly started or restarted participants lies in each participant providing an adequate RNG shortly after starting.[24,25,26]

4.5 SAK installation and use

The effort required from any participant to install a fresh SAK for reception and transmission can be divided into the following broad categories:

a) Unwrapping the received AES Key Wrapped SAK, extracted from a Distributed SAK parameter set.

b) Calculation of any intermediate tables required for Cipher Suite processing with that particular SAK.

c) Creation of the appropriate SCI and AN (Association Number) indexed tables for reception, including per SC counts and the per SA lowest acceptable PN record (see 10.6 and Figure 10-4 of 802.1AE-2018).

d) Calculation of any additional Cipher Suite-dependent values applicable per received SA (SCI and AN indexed) and per transmit AN (such as the SSCI for the XPN Cipher Suites) or applicable to all SAs for that SAK (such as the Salt for the XPN Cipher Suites).

updating the appropriate hardware or memory structures for b) through d).

Of the above a) and b) can be expected to be the most work, although for CAs with a very large number of participants c) and d) might not be negligible.

[22] The exact 9.8 rules should be further spelled out in any future revision of 802.1X, as the current text might be carelessly misinterpreted by the hopeful. Prohibited redistribution includes, of course, any SAK previously obtained from a different Key Server by the current Key Server. The current text does address the possibility of a server-swapping attack on a participant.

[23] See 6.2, 9.2.1, 9.3.1, 9.3.3, and 9.8.1 of 802.1X for general Random Number Generator (RNG) requirements, and 9.4.2–9.4.4, 9.8, 9.10, 9.17, 9.18.3, 9.18.4, 9.19, and 12.2, for MI generation and use.

[24] As a practical matter, the acceptable probability of prior MI duplication (where it differs from the ideal) may need to accommodate the deployment of a very large number of instances of the basic design while an attacker could benefit from a small number of successful attacks.

[25] The potential technical background reading list is extensive and I have decided not to speculate further on potential approaches, particularly for very low cost participants, in this note.

[26] See 6.2, 9.2.1, 9.3.1, 9.3.3, and 9.8.1 of 802.1X for general Random Number Generator (RNG) requirements, and 9.4.2–9.4.4, 9.8, 9.10, 9.17,Review 9.18.3, 9.18.4, 9.19, and 12.2, for MI generation and use.

5. Rapid Group CA formation

5.1 Joining together

The protocol exchanges illustrated in Figure 1 and Figure 2, and described in the standard, are not the only possibilities. Participants do not have to join an existing CA one by one. In fact the existing constraints on the rate of fresh SAK distribution exclude that possibility when initial MKPDUs from several participants arrive in a brief interval, as described in 4.1 above. Figure 3 shows a scenario in which the Key Server has delayed SAK distribution to Alice long enough to receive an initial message from Bob as well, so the initial SAK is distributed to both Alice and Bob.

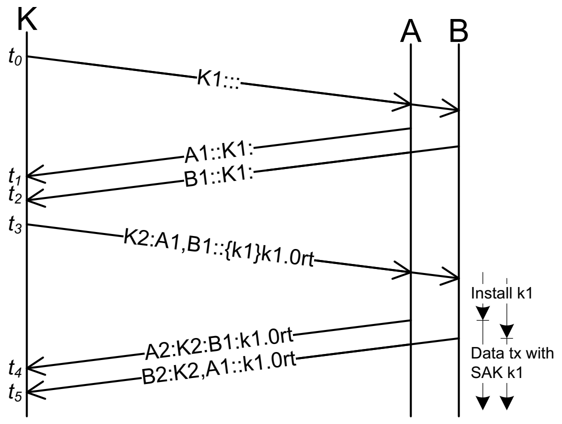

Figure 3—Three initial participants

Additional participants could also feature in this scenario, each transmitting the same initial MKPDU from Kevin, and each transmitting its own MKPDU to prove liveness to Kevin, before receiving the initial SAK and installing it for both transmission and reception (the reception, by Alice and Bob, of each other's initial transmissions so they have the necessary SCI information is not shown in the figure). Only one additional MKPDU need be transmitted for each additional participant in this fortunate scenario (the last pair of MKPDUs shown in the figure only serve to advertise Alice and Bob's current state).

Similarly, after an initial SAK has been distributed to Alice, following participants Bob and Charlie might join together (as existing constraints on SAK distribution rate might require), as shown in Figure 4.

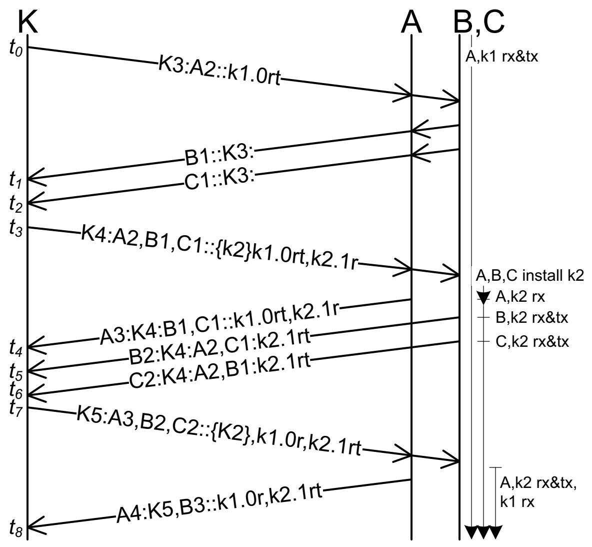

Figure 4—Two join together

5.2 Prior knowledge

Rapid installation of SAKs by all the intended CA participants can thus benefit from appropriate Key Server SAK distribution timing. In particular, if the challenge is that their availability is likely to be roughly but not exactly synchronized by power supply availability it helps if the Key Server has some idea of the target time for full CA operation and:

a) The maximum expected time between Key Server availability and the last participant becoming available; or

b) The expected number of participants for viable system operation following establishment of secure connectivity; or

c) The identity (MAC Address) of each of the essential participants.

With the last of these being obviously the most useful. The description of MKA operation in the 802.1X standard does not assume that the CA participants have any prior knowledge of each other, essentially discovering their identities and the fact they can communicate after an authentication protocol has provided each with the same CAK (and authorization data), or after some physical connectivity has been established, or after power up. However, in some potential applications those participants have been previously physically installed and software configured in a more or less fixed network.

The constraints upon such participants may include a limited ability to record and recall any data that changes from power on to power on. In the interests of cost most might have no non-volatile memory that is capable of being modified each and every time they power up.

Some participants might have a record of the identities[27] of the others with whom they need to cooperate, either as part of their fixed code or install time written memory, while others might have a very limited record of their environment, even though they have experienced it many times and their designers hope is that they will continue to do so. In the latter case, the Key Server system can be responsible for orchestrating the operation of the networked participants, and the fact that MKA provides a secure fully distributed multipoint-to-multipoint transport between authenticated participants and is not limited to activating MACsec should not be overlooked. While the amount of data MKA can distribute is limited by the size of MKPDUs (at least), and is not an efficient substitute for MACsec protected communication, it can distribute EAPOL announcement TLVs including Organizationally Specific TLVs (see Figure 11-15, 11.12, Table 11-8 of 802.1X-2020, and Annex D of IEEE Std 802.1Q for examples) if configuration prior to MACsec operation is required. One possibility is for the Key Server to supply its already known list of SCIs even in advance of the users of those SCIs having completed power up (MIs are not required to support installation of SCIs for reception). Supplying such a list does not preclude later dynamic addition of new participants.

5.3 SAK pre-distribution

A participant can expend a significant part of the effort required use a fresh SAK [a) and b) of 4.5 above] before enabling protection and validation of data frames [which might be only part of 4.5 c) and d)]. Since the initial key unwrapping and key installation stages have no externally visible consequences, a participant could undertaken them before the Key Server decides (by transmitting a Distributed SAK parameter set and a Live Peer List in an MKPDU) on the set of participants that are to use an initial SAK.

An SAK could be ‘pre-distributed’ (as described below) even before the Key Server becomes aware of their active presence. The decision to act on such a ‘pre-distribution’ lies with individual participants, and choice to do so (or not) depends on the expected level of exposure to DoS attack in the scenario for which the participant has been developed, as follows.

If the potential CA concerned is thought to be both (a) somewhat isolated, only exposed thorough the prior installation of a rogue component, and (b) amenable to rapid mission suspension, followed by out of operation diagnosis, then an initial SAK (protected by the Key Wrap specified for MKA) might even be transmitted in an entirely separate message (not an MKPDU), avoiding the requirement for the Key Server to calculate a CAK dependent ICV for that message. A protocol design choice to use such an unprotected message does however balance the cost of ICV protection against the probable need to transmit more messages in total, as MKPDUs will also be required at some stage to establish participant liveness. An optimal design decision depends on accurate cost figures, so, while bearing the possible use of unprotected messages in mind, the following proposal is based on the use of MKPDUs.

Pre-distribution adds two new MKPDU parameter sets, one with the existing Distributed SAK parameters for GCM-AES-128, and the other with those for explicitly identified Cipher Suites. These new sets do not couple the transmission of their parameters to their immediate installation and use based on the Live Peer List in the same MKPDU.[28]

The addition of these new parameter sets forces an MKA version increment. At the same time a previously reserved bit in the existing Distributed SAK parameter set can be used to signal that participants should not begin transmission until they receive an MKPDU indicating that the Key Server has started transmission with the new SAK, allowing the Key Server to delay initial transmissions until it knows that all are ready to receive.

Figure 5 shows a possible start up sequence. The notation ‘[k1]’ represents a key wrapped SAK, with Key Number 1, pre-distributed by Kevin starting in an MKPDU transmitted at t_0. Alice and Bob receive that MKPDU and can start key installation, but cannot yet use k1 to receive data frames as they have as yet no proof of the liveness of Kevin or of the data they might receive—both might be replays of past messages. They both respond with Kevin’s MI.MN, establishing their liveness, allowing Kevin to transmit a Distributed SAK ‘{k1}’ with a Live Peer List which selects the participants that are allowed to transmit using that SAK. Alice and Bob now know they are participating

[27] Such identity information might include the assignment of well-known (from the point of view of running code) local MAC addresses of other participants.
[28] Changing the semantics of the existing parameter sets would be an unacceptable violation of the EAPOL versioning rules.

in a live exchange, check the octets of the key wrapped SAK with those pre-distributed to confirm that they do not need to repeat the unwrapping and installation process, begin receiving with that key and transmit MKPDUs confirming that they can receive protected data frames from Kevin and all participants on the Live Peer List accompanying the Distributed SAK. They will start transmission following receipt of Keven’s next message, which confirms that all participants have reported their reception status (or that Kevin has decided to proceed with transmission without that confirmation).

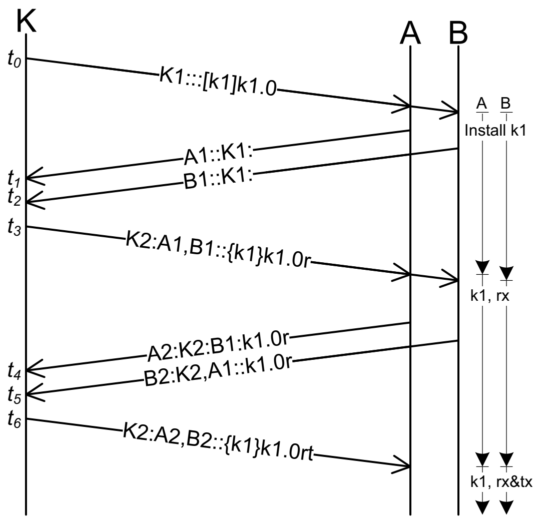

Figure 5—Pre-distribution

If a lossless initial data transmission is not required, Kevin can start transmitting when sending the MKPDU at t_3, with Alice and Bob beginning transmission on its receipt.

Figure 6 shows another possible start up sequence. Bob arrives too late to be included in Kevin’s initial pre-distribution, but responds to that MKPDU, proving liveness and is then included in Kevin’s subsequent SAK distribution. Kevin could of course continue to pre-distribute, with the possibility of adding further participants before transmitting a Distributed SAK parameter set for initial protected data transmission using that key.

In some applications, where the availability of new participants is roughly synchronized by some event such as power availability (see 3 and 4.1 above), a potential participant might restart during the process of initial MKPDU exchange, using the same SCI but with an incomplete record of its prior state and thus of necessity (4.4 above) using a fresh MI. In Figure 6 the participant temporarily identified as Bob reappears as Charlie. Since Bob was not include in Kevin’s Live Peer List in an MKPDU with a Distributed SAK parameter set, Kevin can replace Bob with Charlie in the transmitted Live Peer List that conveys that parameter set.

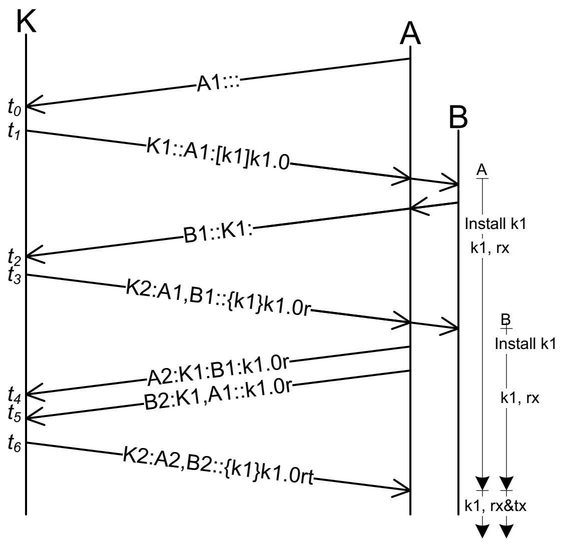

Figure 6—Pre-distribution (2)

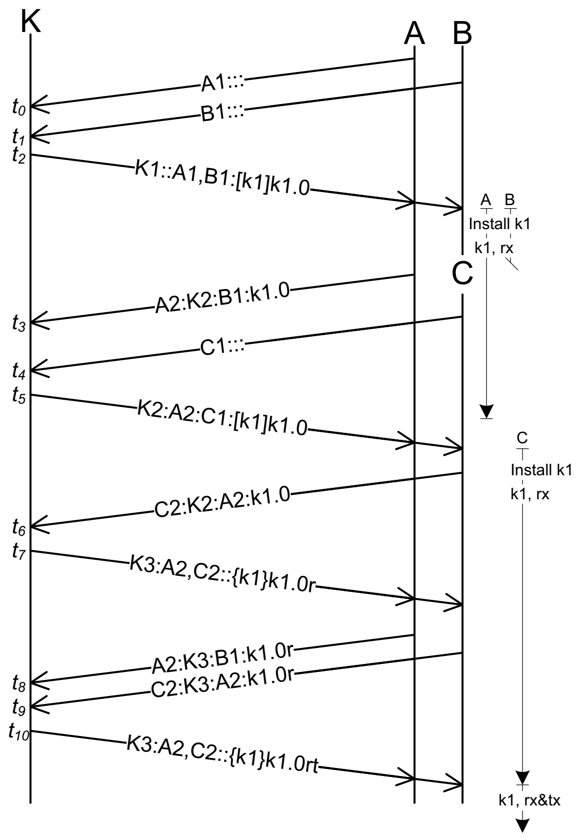

Figure 7—A participant restarts

That replacement does depend on Kevin's ordered reception of MKPDUs. In the worst case the past memory of Bob might have to be timed out before Charlie can be included following SAK rollover.

6. Continued SAK distribution

A fresh SAK is distributed whenever the Key Server's Live List changes (9.8 of 802.1X). This provides a Cipher Suite independent defence against nonce reuse—a participant that resets, forgetting its prior PN use and restarting its PN sequence with the next SAK it receives, is also obliged to forget its prior MI.[29,30]

When one of the current non-XPN Cipher Suites is being uses, the SCI (a concatenation of each participant's MAC Address and port number)[31] divides the nonce space between participants. So the rule forcing fresh SAK distribution could be relaxed for Key Servers that retain a complete record of {MI,SCI} tuples for the SAK currently being distributed: new participants need only force fresh SAK distribution only if their SCI was previously used with a different MI. That should lessen the load for participants that have already installed a current SAK. Additionally such an existing participant need only validate and respond once a second or so to a stream of successive MKPDUs from the same Key Server that only serve to convey the SAK to new participants. Those periodic responses will suffice to retain its presence on the Key Servers Live List.

While this (6) optimization does not involve any change or addition to the existing MKPDU format and TLVs, it should be subject to the scrutiny and documentation that comes with standardization—verifying that it does indeed address a real need not met by the existing standard or optimizations previously described, that envisaged use cases do not require fresh SAKs for other reasons, and ensuring that it not used with any competing optimizations that might also be thought to be possible with the existing MKPDU specification.

6.1 Lossless considerations

A potential drawback to continued SAK distribution, as describe immediately above, is the absence of any convenient mechanism for informing each new participant(s) that the existing participants can now receive frames transmitted by that new participant, and (vice versa) of informing each of the existing participants of when the new participant can receive a data frame that they might transmit. Tweaking timers at each and every participant is unsatisfactory and having new data frames transmitted at very rapid intervals to minimize the delay in initiating communication might also be unsatisfactory. One way of addressing this issue (if it matters) is to use the existing rollover procedure, with a new Key Number and a 'nonce extension' as described in 6.3–6.7 below—substituting the distribution of a fresh 'SAK+nonce space' for the distribution of fresh SAK.

6.2 Participant restarts

Continued SAK distribution as described above (6) reduces the load placed on existing participants (by not requiring that they install a further SAK) as new participants (with a distinct SCI) are recognized as Live by the Key Server. It does, however, require fresh SAK distribution if a participant already on the Key Server's Live List restarts with the same SCI. Simple continued SAK distribution with the same Key Number also does not support XPN Cipher Suites, as Live List additions and removals can change SSCI assignments (see 9.10 of 802.1X). However nonce extension could be used to reduce the effort required to use a new SAK.

6.3 Nonce extensions

The requirement for fresh SAK distribution stems from the absolute need to avoid nonce reuse with the standardized Cipher Suites. Distribution of an SAK, as specified by 802.1X-2020 allows a participant to use that SAK together with the participant's SCI and a 32-bit PN (for non-XPN Cipher Suites) or with the participant's SSCI and a 64-bit PN (for XPN Cipher Suites). The same SAK could be used with a repeated PN without reusing a given {SAK, nonce} if other fields extend the nonce values.

The details of possible nonce extension techniques are Cipher Suite and procedure specific. The following descriptions begin with the use of the existing XPN Cipher Suites together with Key Number rollover: a combination that does not require changes to the way those Cipher Suites are currently standardized, so can be orchestrated by the Key Server without changes to

[29] The reset participant (A, say) will only accept an SAK from a Key Server (K) when its (A's) new MI has appeared on the K's Live List, which will have caused K to distribute a fresh SAK. K cannot reliably track and update A's PN use, as [in the threat model, a) in 9.1 of 802.1X] the attacker could have selectively limited the propagation of A's frames.

[30] A further use case specific consideration concerns possible theft of a participant system and extraction of the SAK. While the CAK and its derived keys that are used to protect and validate MKPDUs might be retained within a secure boundary in the system, it is most unlikely that such precautions could be applied to use of the SAK. There is no suggestion that SAK changes provide perfect forward secrecy (PFS), but it could raise the cost of some attacks.

[31] In most cases each port (physical MAC entity) will have its own MAC Address, so the port number component will not play a significant role.

other participants, and has the desirable lossless characteristics for existing, new, and restarted CA participants.

A similar procedure can be used for the existing non-XPN Cipher Suites. As described this adds bits from the Key Number space into the existing SCI Port ID field, so should be usable with existing MACsec protection and validation hardware. A variant of this procedure avoids key rollover (minimizing the effort required from existing CA participants) at the expense of lossless additions for new participants.

Finally, the syntax and semantics of the Live Peer List can be tweaked to support minimal effort additions and restarts, again at the expense of lossless additions.

6.4 XPN rollover

When the XPN Cipher Suites, (GCM-AES-XPN-128 or GCM-AES-XPN-256) are used, a 96-bit Salt (10.7.27, 10.7.28, 14.7 and 14.8 of 802.1AE) is XOR'd with each participant's 32-bit SSCI and 64-bit XPN to yield the Cipher Suite nonce.

When MKA is used for key agreement, the 64 least significant bits of the Salt are the 64 least significant bits of the Key Server's MI, the 16 next most significant bits of the Salt are the 16 next most significant bits of the Key-Server's MI XOR'd with the 16 most significant bits of the MKA Key Number (KN). The 16 most significant bits of the Salt are the 16 most significant bits of the Key Server's MI XOR'd with the 16 least significant bits of the Key Number.

The Salt thus creates a distinct fraction of the nonce space for each of the first 2^{16} Key Numbers used with its MI.[32] MKPDU size limits ensure that all SSCIs will be encoded in the 16 least significant bits of the SSCI field, and will not interfere with this {Key Number.MI bits} space in the Salt.

So the Key Server could, when using these XPN Cipher Suites, continue to distribute the same SAK even though the Key Server's Live List (listing MIs of participants allowed to use the SAK for transmission) has changed provided that the Key Number is changed with each change in the Live List. That Key Number change prompts the necessary SA AN (Association Number) increment, and the SAK installation and rollover procedures (see newSAK in the CP state machine, Figure 12-2 of 802.1X) that accompany distribution of an SAK with a new Key Number.

A participant that receives a distributed SAK with a new Key Number can skip the calculations necessary to unwrap and install the SAK if a simple string comparison with the prior wrapped SAK shows it to be a repeat. The fresh set of SSCIs do need to be installed, and the ability to receive using this new SA reported (using the MACsec SAK Use parameter set, Figure 11-10 of 802.1X) so the Key Server can coordinate transmit rollover for all participants (transition from CP:READY to CP:TRANSMIT in the CP state machine, Figure 12-2 of 802.1X).

Changing Key Number without changing the SAK when the XPN Cipher Suites are being used, as described here (6.4), does not require changes or additions to the existing MKPDU and TLV formats. Indeed existing implementations that are unaware of the potential optimization and do not check for the repeated SAK will (if correct) interoperate with Key Servers and other participants that use it. However this optimization does not strictly follow all the rules for fresh SAK use specified in 9.8 of 802.1X, so should be standardized, and the comments in 1 regarding public scrutiny of both need and mechanism apply.

While the upper, Key Number influenced, bits of the Salt could be used to allow a given SAK to protect more than 2^{64} frames, that is not the intent of this optimization, which rather addresses reducing the workload during periods of significant change. Even at 1 Tb/s fewer than 2^{40} back to back minimum sized Ethernet frames can be transmitted in a week, and cryptanalytic attack should not be made easier by unnecessarily prolonging use of a single SAK.

6.5 PN nonce extension with KN rollover

When a non-XPN Cipher Suite (GCM-AES-128 or GCM-AES-256) is used, the 64 most significant bits of the 96-bit nonce (IV, 14.5 and 14.6 of 802.1AE-2018) are the octets of the SCI. The most-significant octets of the SCI are a MAC Address associated with the transmitter and the two least significant octets are a Port Identifier. The inclusion of the Port Identifier supports the following possibilities: a single system MAC Address could be used for multiple physical ports (MAC entities), although current standards recommend or require each to have its own unique MAC address; or multiple virtual instantiations (as yet unspecified) of the physical port could be supported in a single CA; or a 64-bit MAC Address could be used without requiring a change to the length of the SCI or its encoding in the MACsec SecTAG (a possibility that diminished for other reasons since the initial standardization of MACsec). Some or all of the Port Identifier bits could thus be used to identify allocate successive fragments of the

[32] In addition to the less significant effects (for this note) of using a Key Server instantiation dependent fraction of the 80 least significant bits of the nonce (IV) for any given SAK value, and not using nonces is strict numerical order.

nonce space in a similar way to that described for the XPN Cipher Suites (6.4). Again the reason for doing so is to lessen the processing load (and consequent delays involved) in repeated SAK installation following closely staggered recognition.

The fragments of nonce space could be identified by encoding least significant bits of the Key Number in the most significant bits of the SCI Port Identifier space.[33] Coupling the nonce space fragments to Key Numbers allows their use to be coordinated by the normal SAK Rollover procedures, just as for the XPN Cipher Suites.

802.1Q specifies 12-bit Bridge Port Numbers. Assuming other participant systems will have modest port counts, this leaves 4 bits to identify alternate nonce spaces. That is probably sufficient, allowing the same SAK to be retained across 16 group formation episodes, during each of which one or more participants could join the Key Server's Live List. The true (i.e. without any included Key Number bits) Port Identifier need to be advertised by each participant in its Basic parameter set (Figure 11-8 of 802.1X) SCI.

Each participant needs to advertise its ability to use this PN nonce extension, and the Key Server needs to be able to select its use for any given distributed SAK and to be able to distribute a SAK that is not to be used with the extension if unsupported by one or more members of the Live Peer List using the SAK. One way to do that would be to assign one of the currently reserved bits in the third octet of the Live Peer List and Potential Peer List to signal support of the capability, and to assign one of the reserved bits in the second or third octet of the Distributed SAK Parameter set to select its use. An MKA Version Number of 4 or above (it is currently 3) would be used by any participant capable of setting either of these bits (see versioning rules in the third paragraph of 11.11 of 802.1X).[34] An alternative approach, consistent with the existing specification, would be to assign two additional MACsec Cipher Suite reference numbers for use in the Distributed SAK Parameter set and in the MACsec Cipher Suites EAPOL-Announcement TLV (type 112 in Table 11-8 and Figure 11-12 of 802.1X). The latter may be thought to be consistent with Cipher Suite specification to date, and would not require a version number increment but does add more octets to the MKPDUs than might be thought desirable. The choice between these approaches does affect the way they are documented.

As described, the Key Number dependent part of the nonce is encoded in the Port Identifier part of the SecTAG SCI to support existing MACsec protection and validation hardware, rather than creating a Salt equivalent (which would be XORed into the relevant bits for protection and validation, and not carried in SecTAG).[35]

6.6 PN nonce extension without KN rollover

The above Port Identifier based nonce extension could also be applied piecemeal, with the Key Server supplying participant specific values. There are a range of possibilities, none particularly appealing at present.

Using additional Port Identifiers [or with the SSCI nonce addition described below (6.7), additional SSCIs] to avoid the KN increment with its need to update the AN [and with XPN the need to update the Salt], and its need to use the CP state machine SAK rollover procedure, also means that the signalling that is part of that SAK update procedure does not take place. As described above (6.1), the Key Server can receive indications that new participants can receive and transmit with the assigned Port Identifiers [or SSCIs] and the existing SAK, but can see no change in existing participants status. So a new participant can only tell when other new participants are capable of receiving the frames it will transmit if it spots their receive status transition in the MKPDUs they transmit, and not by receiving information in Key Server MKPDUs. A new participant has no way of telling when existing participants have installed its Port Identifier [SSCI]. Existing participants also have no way of knowing when the new participant is capable of transmission and reception, as far as they are concerned the MAC was already operational (OperUp) and continues to be OperUp.

[33] Other solution are of course possible, including creating new parameter sets, extending existing parameter sets, and borrowing reserved bits from existing fields including (notably) the Live Peer List and Potential Peer List parameter sets —at least one of these parameter sets needs to be present in MKPDUs sent by participants and by the Key Server prior to SAK distribution to Live Peers.

[34] Not all future versions of MKA should be tied to this capability, so a version number increment alone is insufficient.

[35] Changing MACsec itself, creating divergent implementations that are supposedly lower cost in certain applications, is more likely to raise costs for both customers and suppliers across the board long term, and constrain future possibilities. Just for the sake of avoiding some future claim of 'invention', I think it is clear that an XPN SSCI could be carried in part of the Port Identifier space (for example) which would 'work' for modest CA sizes and MKA, but close out future flexibility with other possible key distribution schemes. The contribution of tweaks such as that to reduction of the overall validation workload is likely negligible, and not worth the additional SKUs.

6.7 SSCI nonce addition

The available nonce space for given SAK can also be effectively extended by simply adding an SSCI when a the Key Server detects a new participant.[36] SSCI assignments can then be defined by the order of their appear in the Live Peer List of the MKPDU that the Key Server uses to distribute a SAK (which may now be a repeat of an earlier distribution). A list MI for a participant that has since restarted (with a fresh MI) or has been timed out, can be replaced with a reserved value (0, for example).[37]

This approach has the appeal of not requiring an MKA Key Number (KN) increment and the accompanying key rollover procedure when the new participant is added. The downsides are the lack, as described in 6.1 and 6.6 above, of support for lossless addition, and the change in the semantics of the Key Server's Live List (as specified in both 802.1AE and in 802.1X) when used in conjunction with Key Distribution. The EAPOL protocol version handling rules (and other similar versioning rules in other 802.1 standards) were designed to prohibit such changes in the interests of backwards and forwards compatibility. The issue and a work around is described in 6.7.1.

If the CA can include a significant number of participants, e.g. 50 or more, and the expectation is that a given participant system might appear as a fresh participant more than once, either due to a potential reboot during an initial erratic power up sequence or due to temporary power down while the overall network of participants continues group CA operation, then SSCI nonce extension will not avoid the need to support SAK rollover as currently required by the 802.1X CP state machine.

6.7.1 Live Peer List semantics

The fourth paragraph of 10.7.13 'Receive SA creation' of 802.1AE specifies SSCI assignment:

"MKA, specified in IEEE Std 802.1X, does not distribute SSCIs explicitly. A KaY assigns SSCI values as follows. The KaY with numerically greatest SCI uses the SSCI value 0x00000001, the KaY with the next to the greatest SCI uses the SSCI value 0x00000002, and so on. This assignment procedure is not necessarily applicable to any other key agreement protocol."[38]

The fifth and subsequent paragraphs of 9.10 of 802.1X-2020 reinforce that ordering, but specify that for MKA Version 3 (or higher) the SSCI assignments for an XPN Cipher Suite are taken from the order in the Live Peer List.

To be certain that the current set of CA participants do not include some that use the original SSCI assignment rule (sequentially, strictly by SCI value) and some that rely entirely on the Live Peer List (sequentially, strictly by list order skipping SSCIs corresponding to reserved entries in the list), the Key Server should be required to use SSCI nonce addition only if all Live Peers are using MKA version 4 (to be safe) or above. Prior version participants could coexist with updated participants in a single CA, but the Key Server would not be able to use simple SSCI nonce addition but would need to use XPN rollover (6.4 above) to add participants, for which support would remain a mandatory part of conformance.

A. Additional background and notes

t.b.s.

[36] Lars Völker identified this possibility, building on the XPN nonce extension previously described (6.4 above).

[37] In theory a random value could be chosen for the replacement, given the extremely low probability of a collision with a real participant value, but that could be confusing both for implementation and administration.

[38] See also the following note.