

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, identifies the participants that are to use it and when they are ready to receive from each of the others, and initiates data transmission protected by it. Participants can or could (with appropriate standardization) reduce the processing required to validate MKPDUs and install keys. This note describes some of what might be done, points out some of the pitfalls.²

1. 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.

The contents of MKPDUs are, very deliberately, not confidentiality protected. This was initially done so protocol operation, and any difficulty in the progress of that operation, could be usefully observed by a network administrator who did not possess the CAK.³ 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 safe, from a security point of view, provided no protocol action is taken as a consequence of processing 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 ().

1.1 Duplicate MI detection

Each participant needs to check the content of MKPDUs transmitted by other participants for duplicate use of its own MI 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.

1.2 Maintaining liveness

MI (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,⁴ risks being aged out by the Key Server and excluded from future communication. So the last unvalidated key Server MKPDU needs to be retained in case it needs to be validated so a response including its MI.MN can be sent.

1.3 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

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 that attacker possesses the CAK (possibly by prior equipment theft).

² 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 IEE, or the 802.1 Working Group or Task Groups.

³ The secure Connectivity Association Key (CAK) is either pre-shared/pre-pace 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).

⁴ 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 is a bet that the next periodic transmission will be received.

MI and recent MN. That confirms a direct, current, 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.

1.4 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 relative effort expended on receiving an MKPDU as opposed its reception and validation. Kevin's transmission strategy also plays a part, as 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.

1.5 Peer SCI 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).⁵

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 [1.4 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.

2. 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

⁵ 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.

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.

3. MKPDU transmission and SAK distribution

The discussion so far suggests lowering the MKA workload for non-Key Server participants by reducing the effort they expend in MKPDU validation. That effort might be considered (by some) excessive in two general cases: (a) when a very large number of participants are involved,⁶ and (b) when very rapid CA⁷ 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. This second case can be addressed, with or without the need to use partial MKPDU validation (as described above), by paying attention to MKPDU transmission timing and the Key Server's choice of when to distribute SAKs.

3.1 Basic MKPDU exchanges

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

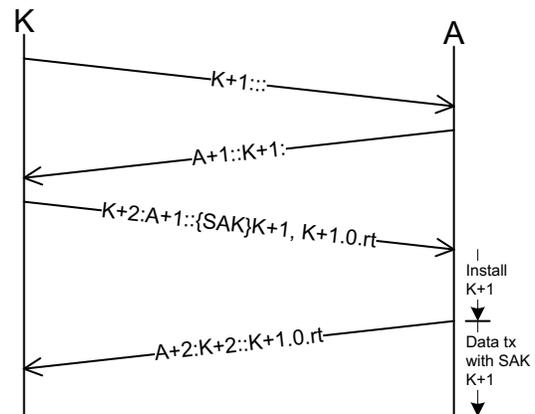


Figure 1—Initial SAK Distribution

Figure 1 begins with an MKPDU transmission, after power up, from Key Server, K, to participant A. Since K and A can complete power up at different times, it is likely that a prior MKPDU has been lost. A first MKPDU from A might also precede the sequence shown, effectively prompting K to begin. The last MKPDU, A+2, merely advertises A's transmit and receive status—its receipt is not a vital part of enabling MACsec-protected communication.

Figure 2 shows a continuation of the dialogue, with a third participant, B, joining. MKPDUs transmitted by B that A does not have to validate (but can validate with lower priority) are shown as dashed arrows. The addition of B to the CA forces (9.8 of 802.1X) the distribution of a fresh SAK (K+2). A has to receive, and validate, two MKPDUs from K—one with the fresh SAK, and one indicating that K has started transmitting use that SAK, so A 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 of these, A transmits an MKPDU when it has installed SAK K+2 for reception, allowing K to transmit the second, coordinating the lossless rollover from K+1 to K+2.

⁶ 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.

⁷ 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.

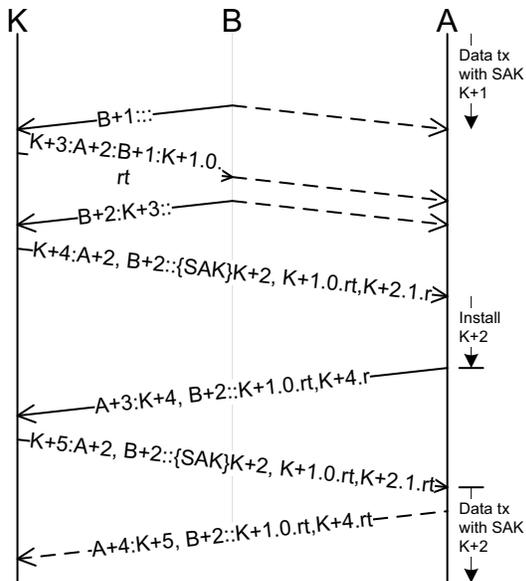


Figure 2—Follow up SAK Distribution

If the addition of participants to a CA is spread over time, the pattern of communication for existing participants on each addition will follow that of A in Figure 2. Each receives a new SAK, installs it and responds, 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 the possibility of 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.

3.2 Rapid Group CA formation

As noted above rapid installation of SAKs by all the intended CA participants can benefit from appropriate 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:

- The maximum expected time between Key Server availability and the last participant becoming available; or
- The expected number of participants for viable system operation following establishment of secure connectivity; or
- The identity (MAC Address) of each of the essential participants.

With the last of these being obviously the most useful. Note that 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 to some extent 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 fact in the interests of cost most may have no non-volatile memory that is capable of being modified each and every time they power up. It is possible that they have a record of the identities⁸ of the other participants with whom they need to cooperate, either as part of their fixed code or install time written memory. It is also possible that most have no such (or

⁸ Such identity information might include the assignment of well-known (from the point of view of running code) local MAC addresses of other participants.

a very limited) record of their environment, even though they have experienced it many times and the designers hope is that they will continue to do so. In that latter case, the Key Server system may be responsible for orchestrating the operation of the networked participants. In that case one possibility that should not be overlooked is that MKA itself provides a secure fully distributed multipoint-to-multipoint transport, and is capable of distributing and activating more than MACsec SAKs. While the amount of data it can distributed is limited by the size of MKPDUs (at least), and cannot be considered 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.

3.2.1 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.⁹

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. Then, 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. When that SAK is then used by the participant with a previously used PN (i.e. repeating Cipher Suite

nonce use) but with different data (assumed to be a result of changes in the participant's environment) the attacker can then use those repeated data frames to recover the SAK and gain receive and transmit access to previously secure communication. Given the aforementioned attack capabilities, other CA participants can be completely unaware of the intrusion.

So, a potential challenge for rapid CA formation, or extension, incorporating newly started or restarted participants lies in each of those participants providing an adequate RNG shortly after starting.^{10,11,12}

3.2.2 Detecting new participants

An important factor in the overall delay from initial Key Server availability is when each of the other participants receives an acceptable (for subsequent liveness proof) Key Server MI.MN. This can be reduced by rapid repeated Key Server MKPDU transmission, either consistently through the start up phase, or in response to an initial transmission from each would be participant. One approach, not addressed in the standard but not requiring any change to the contents of transmitted MKPDUs, is for the Key Server to poll by repeating exactly the same MKPDU with unchanged MI.MN. Provided the overall repeat time is short, this should not significantly reduce the Key Server's ability to timeout inactive participants. However other participants should avoid unnecessary processing of MKPDUs from the Key Server or any other participant (as identified by transmitter MI) by only validating those whose MN is greater than that last processed or awaiting processing. Since each MKPDU reflects the current state of its transmitter (and not just one of a succession of commands) information from the last is all that is required.

3.2.3 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.^{13,14}

When one of the current non-XPB Cipher Suites is being used, the SCI (a concatenation of each participant's MAC Address and port number)¹⁵

⁹ 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.

¹⁰ 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.

¹¹ 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.

¹² 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.

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 (3.2.3) 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.

3.2.4 Participant restarts

Continued SAK distribution as described above (3.2.3) 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. 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).

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 as described below ().

3.2.5 XPN nonce extension

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.¹⁶ 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

¹³ 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.

¹⁴ 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.

¹⁵ 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.

¹⁶ 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 in strict numerical order.

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 (3.2.5), 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 3.2.3 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.

3.2.6 PN nonce extension

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 nonce space in a similar way to that described for the XPN Cipher Suites (3.2.5). 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.¹⁷ 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).¹⁸ 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.

¹⁷ 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.

¹⁸ Not all future versions of MKA should be tied to this capability, so a version number increment alone is insufficient.

The PN nonce extension described here (3.2.6) does require standardization.

3.2.7 SSCI nonce extension

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.¹⁹ This has the appeal of not requiring an MKA Key Number (KN) increment and the accompanying key rollover procedure when the new participant is added. It arguably requires a change in the MACsec standard (802.1AE) and in the semantics of the Key Server's Live List (as specified in 802.1X) when used in conjunction with Key Distribution. Semantic change is what the EAPOL protocol version handling rules (and other similar versioning rules in other 802.1 standards) were designed to prohibit. The issue described further below.

Using additional SSCIs to avoid the KN increment with its need to update the AN, its need to update the XPN 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. The Key Server can receive indications that new participants can receive and transmit with the assigned 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 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.

The issues with allocating SSCIs as participants are detected by the Key Server (allocating a fresh SSCI for each MI, so a rebooted participant will use a different fraction of any given SAKs overall nonce space) are as follows.

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."²⁰

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 from the order in the Live Peer List. That appears to be definite enough to break the tie with the SCI defined order.

The existing 802.1X specification does not have a reserved MI value to indicate that no MI (and hence no SCI or SA) is present for a given SSCI position in the Live Peer List, and one is required to avoid needless creation of a received SA, although if the Key Server picks a value (possibly dynamically in the worst case) only one wasted SA would be created.

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.

A. Additional background and notes

t.b.s.

¹⁹ Lars Voelker identified this possibility, building the XPN nonce extension previously described (3.2.5 above).

²⁰ See also the following note.