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, communicating with all the others and reducing the number of MKPDUs required to add a new participant to an existing group. 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) delay bounding of transmitted data. However 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 other than the Key Server can reduce the processing required to validate MKPDUs transmitted by others. This note describes what can be done, points out some of the pitfalls, and notes related performance optimizations.[1]

1. Selective MKPDU validation

The abstract (above) introduces the basic idea—MKA participants other than an elected or aspiring Key Server can omit, or treat as lower priority, validation of MKPDUs from participants other than the elected Key Server.

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.[2] That avoided any need for a possibly vulnerable management interface for key access. In the context of this note, 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.

2. Live and Potential Peers Lists

An attacker could send MKPDUs that would not pass validation to disrupt MKA operation. Optimizing validation reduces the load generated by reception of those MKPDUs, but potentially confuses operation. MIs (Member Identifiers) should only be added to a participant's Potential Peers or Live Peers List as a consequence of receipt of an MKPDU that passes validation. If the participant has validated only MKPDUs recently (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. A network administrator needs to be aware of the reason for the absence of other entries on the participants lists, and for this reason if no other it would be advisable to standardize the optimization if it is 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 transmitted by a participant.

3. 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.

4. Received SAs and replay protection

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). Each received MACsec protected frame carries an SAI (Secure Association Identifier, comprising the SCI and a two-bit Association Number, AN) used to identify and update

[1] This note follows up on a brief discussion in the 802.1 Security Task Group, November 2024.

[2] The secure Connectivity Association Key, 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.

the lowest acceptable PN for the SA. 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).[3]

The SCI of each peer is not included in MKPDUs transmitted by the Key Server (both MAC Address and port number components) of each peer is available. The mapping between an MI, transmitted in the Key Server's Live Peer List, and the corresponding peer's SCI is available in MKPDUs transmitted by the peer. 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. However such attacker-transmitted MKPDUs with an MI duplicating that of a valid participant but with a different SCI could be sent by an attacker as part of an DoS attempt, where the attacking system is not connected to the LAN in a way that would allows it to simply interfere with transmission from the valid participant. If a participant sees such a duplicate MI use, it should validate the transmitted MKPDUs before installing an associated SA.

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.

5. 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;[4] and (b) when very rapid CA [5] 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.

5.1 Basic MKPDU exchanges

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

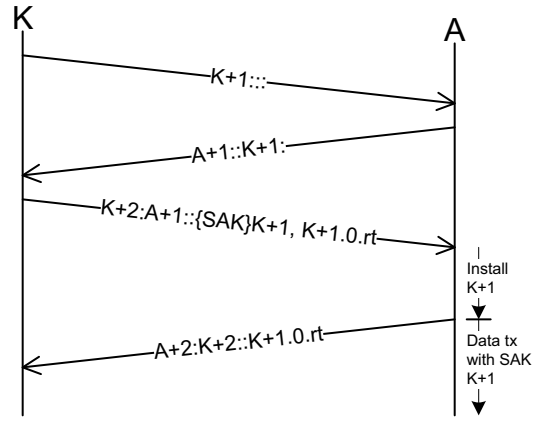

Figure 1—Initial SAK Distribution

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

[3] 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.

[4] 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.

[5] 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.

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.

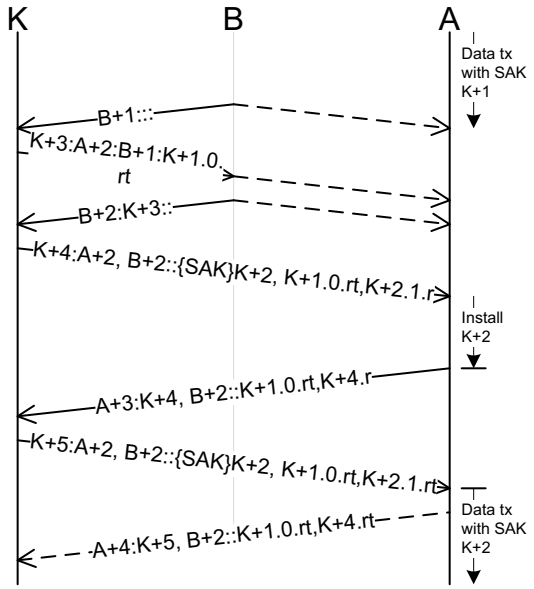

Figure 2—Follow up SAK Distribution

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.

5.2 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:

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.

5.2.1 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, 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.[6]

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

[6] 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.

(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.[7,8,9]

5.2.2 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.

5.2.3 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.[10,11]

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)[12] 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 (5.2.3) 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.

5.2.4 Participant restarts

Continued SAK distribution as described above (5.2.3) 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

[7] 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.

[8] 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.

[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] 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.

[11] 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.

[12] 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.

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 ().

5.2.5 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.[13] 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 (5.2.5), 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 5.2.3 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.

5.2.6 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

[13] 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.

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 (5.2.5). 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.[14] 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).[15] 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.

The PN nonce extension described here (5.2.6) does require standardization.

A. Additional background and notes

t.b.s.

[14] 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.

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