A distributed fault-tolerant group key selection protocol for MACsec

Mick Seaman

This note proposes a Key Selection Protocol (KSP) that allows MACsec participants to prove mutual authentication through common possession of a master key, agree on SAKs, and ensure that data traffic and keys are neither being delayed nor replayed. KSP also provides a simple unreliable transport for protocols that provide new master keys.

The problem to be solved

KSP[1] addresses a slightly different problem from that commonly assumed for group key management for internet applications.

Most significantly, since MACsec is designed to underpin basic network connectivity, it cannot be assumed that an authentication server is accessible whenever fresh session keys are required. MACsec and KSP therefore use pre-shared master keys. At the same time it is often desirable to support MACsec with a AAA infrastructure, so KSP facilitates the acquisition fresh master keys whenever the necessary resources are available.

MACsec:

- provides security primarily to protect the network infrastructure, allowing spanning tree, routing protocols, and DHCP[2] to be protected, rather than providing end to end security for applications
- operates over a single LAN, so there are no intervening bridges or routers[3], and multicast transmission is usually as cheap as unicast
- places a premium on getting up and running rapidly, lest network users conclude that there is no network at all
- needs to be supported with little computation, otherwise network stability may be threatened
- uses master keys that are likely to be already one or two steps removed from authentication credentials
- connects only those stations attached to a single LAN[4], orders of magnitude fewer than envisaged for internet multicast applications

These factors admit a different, and hopefully easier solution.

Overview

This note begins with a few of KSP's goals, reviews what MACsec does, and proposes that the stations that compose a MACsec secure connectivity association (CA) use a common master key (CAK) as the token of their mutual authentication and authorization.

All KSP messages are protected with the CAK, and verification of an integrity check value (ICV) allows each receive to confirm that the transmitter possessed that key. However the CAK cannot be used to protect MACsec data directly, as that could lead to the use of repeated key and nonce values[5]. To ensure this does not happen, each station insists on using a fresh secure association key (SAK) to protect data whenever it is reinitialized.

To prevent a compromised CAK leading to disclosure of a SAK, the latter is actually composed from a 'key number' (KN) distributed by KSP and a 'key generating key' (KGK) that is not. The latter is distributed or installed along with the CAK[6].

KSP ensures the liveness of its messages so that an attacker cannot replay them to force repeated key use. Each station generates a random identifier whenever it is renitialized[7] and maintains and publishes an age for that instance. A proposal for a new KN is not accepted unless contained in a message that includes the receiver's identifier with a satisfactorily recent age.

An age is also associated with each KN. This is used to ensure that the data delivered by MACsec has been neither replayed nor delayed

[1] There seems to be fair agreement on what Key Agreement, Key Exchange, and Key Distribution Protocols do. KSP differs in some respects from all of these, hence the new name.

[2] Among other protocols that underpin IP.

[3] At least none that are visible to the user of the LAN service.

[4] Or virtual LAN supported by a switched infrastructure.

[5] Which is expressly forbidden for GCM, GMAC, and other modes that do not use a purely random nonce for each data packet.

[6] KSP keeps the key number confidential, though this is really unnecessary. This rigmarole is included to reduce the urge to include public key cryptography computations in KSP. If the KGK is determined by a DH exchange along with the CAK there is no reason why it should ever be transmitted or used to encrypt a message. It cannot be discovered by cracking the SAKs it helps to generate as these are the result of a one way hash. EAP distributes enough bits to supply both the CAK and the KGK.

[7] i.e. whenever it forgets its previous instance identifier.

for more than two seconds[8]. Each station regularly advertises its recent transmit KNs, each with its age and the oldest packet number to be accepted by the receiver.

KSP supports key sharing[9] so the number of simultaneous SAKs does not increase with the number of stations. Each message contains a 'proposed key number' (PKN), each station adopting the suggestion made by the highest priority live station. The suggested key also has an age, and stations can request a newer key. The rate at which the suggested key is changed is limited, thus allowing stations in the CA to follow key changes.

Goals[10]

KSP and the assumptions made about its placement within the overall key hierarchy for a network, attempt the following[11]:

1. Maximize the chance that the required full, i.e. symmetric and transitive, connectivity is provided between stations on the LAN.
2. Provide secure connectivity within a few seconds of the underlying LAN service becoming available.
3. Function correctly in an environment where stations are powered on and off at any time, and where an attacker may control power.
4. Allow any subset of stations to be powered off and on again without disrupting the connectivity[12] between the remainder of the stations.
5. Support both point-to-point and multipoint connectivity without preselection of one or the other.
6. Ensure delay bounds for MACsec data traffic.
7. Operate without requiring pairwise communication between all stations.
8. Allow connectivity to be re-established after power-up without requiring network connectivity to an authentication server.
9. Protects against attacks that attempt to exhaust resources by requiring difficult cryptographic calculations.
10. Minimize the number of SAKs that each station needs to support at any one time.
11. Operate without the use of computationally expensive public key cryptography techniques.

Non-goals

1. Guard against or compensate for the use of weak keys[13].

What MACSec does

MACsec secures a LAN. In many cases this means a physical point-to-point link. In others a number of LAN equivalents may be realized by multiplexing over a physically shared media. In another a virtual LAN may be provided at many customer sites by a provider bridged network. In all these cases the service provided by MACsec is that used by bridges and end stations. Securing a LAN is different from securing individual connections amongst stations connected to that LAN[14]. All that MACsec guarantees is that integrity and confidentiality will be preserved amongst the set of stations authorized to connect to the LAN.

An important goal of MACsec is not to change the way that bridges and routers work, while enabling them to apply policies[15] to the data that they forward. This means that MACsec itself should not interfere with the normal connectivity provided by a LAN to authorized stations.

Connectivity Associations

The authorized stations that are attached to a common LAN compose a MACsec secure connectivity association (CA), and prove their mutual authentication by exchanging messages that are integrity protected with a common master key, the CA Key (CAK).

The CAK may configured in each station out-of-band, using a local command line interface for example. Alternatively the CAK may be the direct or indirect result of executing a key agreement, key exchange, or authentication and authorization protocol. If the CA is point-to-point (i.e. has only 2 members) , the CAK may be the

[8] Frames delayed more than ½ second may be discarded by a LAN service, frames delayed more than 2 seconds must be. Ideally the replay and delay window would be tightened to match the acceptable media queuing and transmission delay bounds, allowing MACsec secured protocols to make the same assumptions as protocols designed in the absence of security considerations. This effectively rules out pairwise challenge-response liveness protocols for even a modest number of stations.

[9] The SCI, different for each station, is part of the MACsec nonce.

[10] Goals are those things that one would like to do and can do, the other things are non-goals. Requirements (in standards development) are things the other guy can't do.

[11] Amongst numerous other goals.

[12] It is not even necessary to change SAKs if each station has a real time clock of even modest accuracy.

[13] Possession of even a single KSP message allows an attacker to attempt an offline brute force attack. The KSP message is integrity protected so the attacker knows with reasonable certainty when the key has been guessed. Possession of a second KSP message confirms the key to a high probability. For this reason KSP is not used with easily guessable password based CAKs.

[14] If this is what is wanted from MACsec, then separately secured LANs need to be provided and interconnected with a trusted bridge or router.

[15] The policies should be based on the authorization accorded to the stations connected to each LAN

pair-wise master key (PMK) generated by EAP with one of the members as the EAP peer and the other as the authenticator. If the CA comprises three or more systems with a full or partial mesh of PMKs, a CAK can be assigned and distributed to each of the members using a trivial spanning tree protocol.

Since MACsec secures full (i.e. symmetric and transitive) connectivity between the members of a CA using symmetric key cryptography[16], all the members of the CA possess all the secure association keys (SAKs) used to support the CA. So the requirement for transitive connectivity ends by implying transitive trust within the CA – if A trusts B and B trusts C, then A necessarily trusts C. This transitive trust is captured by the single CAK. An attempt to enforce partial connectivity to match a partial mesh of master keys would cause client protocols to behave oddly, if not incorrectly. Equally the direct use of a mesh of keys opens up the prospect of complex accidental failures.

Secure Channels

In order to provide unique cryptographic nonces and replay protection, the data traffic from each transmitter in an CA is identified as belonging to a separate secure channel (SC). Each SC is supported by a succession of secure associations (SAs). One SA is replaced by another with a different secure association key (SAK) when the packet number (PN) space for an SA is close to exhaustion, or when the CAK is changed. SAKs may be unique to an SC, or shared amongst some or all SCs in the CA. KSP shares SAKs so that each participant in a CA only has to be able to receive and transmit using the same number of cryptographic keys as required for a point-to-point CA.

Changing Keys

KSP allows both SAKs and CAKs to be changed without disrupting the connectivity between stations, although one reason for changing a CAK is to create a new CA that excludes members of a prior CA.

While addressing the requirements of multipoint CAs, KSP is simple enough to be used unchanged if the CA is only point-to-point. This maximizes interoperability and helps considerably in those cases where a CA is initially believed to be point-to-point but turns out to be multipoint. KSP has to work well in an environment where stations are being powered up and down and different systems take more or less time to become functional after power up.

Station priority

At times a decision has to be made between otherwise equally acceptable keys proposed by two or more stations. Each station's priority comprises its SCI and a 'member operational' (MO) indication which is set if the secure service provided by the station is MAC_Operational. To minimize disruptive key changes, proposals from stations with MO set have higher priority. If two proposals have the same MO, then the lower value of the SCI is preferred.

The need to agree SAKs

Why not simply use the CAK as the one and only SAK, and avoid the need for any protocol?

Apart from the requirement to extend the PN of every MACsec data frame to allow the SAK to be effectively used for ever, every system would have to remember how much of the nonce space it had used so far. Not only would this information have to be kept across power outages, it would mean that reinitializing a system with a PMK and SCI combination that had ever been used before would have to be prevented, or the highest PN ever used by a system with that combination would have to be known. This is extraordinarily difficult to achieve, and is highly likely to be circumvented by network administrators replacing or reinitializing failed systems. The alternative of making the nonce for each and every packet random in a space so large that duplicates are vanishingly likely to occur would add a large number of octets to each MACsec header, at least 16 and quite possibly 32 or more.

Using a fresh CAK every time a system is powered up is plausible if authentication is to be carried out afresh each time. Since MACsec addresses the needs of infrastructure communications, relying on reauthentication and the possible use of the infrastructure itself to reach centralized authentication services will make for extremely slow recovery after power failure as the infrastructure would have to be rebuilt hop by hop. Since there is no way of communicating this network outage in a useful way to attached hosts, they can be expected to have adopted their own failure recovery strategies in the absence of the network, compounding the effect of the delay.

Use of public key cryptography and local authenticated key exchange protocols to generate a fresh CAK is possible, but imposes a further administrative burden on users, and may be completely unacceptable to those who wish to use centralized management based on AAA protocols.

For all these reasons it is desirable to allow the CAK to be cached across power cycles[17], and essential that the CAK not be used as an SAK or to algorithmically generate SAKs. Instead SAKs

[16] For the picky it also has to be noted that MACsec does not use more than one key to integrity protect a frame, and that each frame handed to MACsec by its user is only sent once.

[17] Provided its lifetime can be enforced with a real time clock.

are randomly generated and distributed and agreed using KSP.

Generating SAKs

KSP communicates sufficient information to allow an arbitrary and changing number of stations to use the same SAK, and to allow the SAK to be updated. KSP does not directly distribute SAKs, but communicates randomly chosen 'key numbers' (KNs)[18]. Each SAK is derived from a KN by an AES-128 encryption of the latter with a 'key generating key' (KGK)[19]. The KGK is installed along with the CAK[20].

Proving liveness

Random choice of KNs from a very large number space is not sufficient to prevent key reuse. An attacker might simply replay one or a sequence of KSP messages. To ensure that the KN or any other information in a received KSP message should be acted on, each station maintains a CA 'member identifier' (MI) and 'member age' (MA). A new MI is chosen, and the MA reset whenever the station cannot recall the MA, or the MA reaches the maximum age supported by the protocol. The MA is incremented at least every ½ second and whenever the station is rebooted, and is never decremented for any given MI[21].

Received KNs, and suggestions that a new KN is required, are not acted upon unless the KSP message contains the receiver's MI with an acceptably recent MA[22].

Responsibility for ensuring this occurs is divided up between the other members of the CA. Each station has to keep a record for each of the other CA members to provide MACsec replay protection. A key accepted bit (KAB) in each record indicates whether the member is known to have accepted the currently proposed key number (PKN, see below), as indicated by the last KSP message received from that station, and is cleared whenever the PKN changes. The member records are kept in a station dependent order[23], and each KSP message contains the MI/MA tuples for the next four records with the KAB bit clear.

While allowing the transmission rate of KSP messages from any given station to be limited to three per second, this allows a CA of several hundred members to be formed within a few seconds of a simultaneous power up[24].

Agreeing KNs

Each station advertises a proposed key number (PKN), and each adopts the PKN suggested by higher priority live stations, subject to acceptability criteria that guard against key reuse. A station does not have to receive the PKN directly from the highest priority station (that it knows about) to begin installing the key. However the PKN is not used for transmission until it has been adopted by the highest priority 'live' station and all live stations have confirmed installation of the key for reception.

The PKN remains the same for as long as possible, to allow all stations to migrate their transmissions to that key, and is usually set to the last key number (LKN) used for transmission, in which case the 'proposing last key' (PLK) indicator is set. The PKN may need to be changed for two reasons.

First, a station can only adopt the key if it can be sure that it will not transmit a frame using a previously used key and PN tuple. Each station remembers the LKN, the highest PN (LPN) used with that key, and guaranteed overestimates and underestimates of its age. The overestimate is used to determine the proposed key age (PKA) when the LKN is being used as the PKN. The underestimate is used to determine the maximum acceptable key age (AKA) for PKNs from other stations, since they may have been used by the station prior to the LKN. If the LKN or the bounding estimates of its age are unavailable for any reason, such as station initialization, the use of that or any key previously held by another station would risk replay. A new PKN is randomly generated, PLK is reset, and both PKA and AKA are set to zero.

Second, a station may be close to exhausting the packet number (PN) space for the key. If PKN identifies the key currently used for transmission, AKA is set to the lower of its current value and half of PKA, a new PKN is randomly generated, PLK is reset, and PKA is set to zero.

Each station continually advertises its PKN, PLK, PKA, and AKA. When a station receives a live KSP message, it checks to see which, if

[18] This aspect of KSP is orthogonal to its other capabilities.

[19] An alternative is to GMAC a fixed 128-bit value, say the CAK, using the KGK as the key and the KN as the IV. The resulting 128-bit ICV is used as the SAK. This approach is a little more elaborate, and doesn't add any value.

[20] In some environments it may be useful to separate the KGK process from the CAK process. For example EAP might distribute PMKs/CAKs to a small population of systems with an out of band configured KGK. Any weakness in the authenticating systems supporting EAP would not automatically disclose SAKs. In this case it would be useful to derive the CAK from the PMK using the KGK or a similar private value.

[21] Which is why the MI is distinct from the SCI, which permanently identifies the station. The MA is not calculated using NTP or any system time which can be run backward.

[22] This requirement can be relaxed so that the receiver only has to see an MI/MA tuple that is an acceptable match for any station that has itself matched the MI/MA of the receiver.

[23] The order is determined by xoring the stations MI with the MI in each record and ordering on the result from low to high.

[24] The optimistic case is that each station has to participate in no more than two liveness exchanges. Since trust within the group is transitive this is capable of starting a CA of arbitrary size. However there is no certain way of dividing up responsibility to achieve this.

either, of the two proposed key numbers will be acceptable to both CA members, updating the information it holds so that the basis for that acceptability is also correctly conveyed in subsequent messages sent to other stations. In the following description each of the parameters held by the station is prefixed by 's.' and those extracted from the received message by 'r.'

1. If r.PKN equals s.PKN, r.PKA and s.PKA are set to the greater of their values
2. r.AKA and s.AKA are set to the lower of their values
3. If r.PKN is not equal to s.PKN, or r.PLK and s.PLK are not both set, then both r.PLK and s.PLK are cleared.
4. If, for either r. or s., PKA is greater than AKA and PLK is clear, that PKN is excluded from consideration.

If either PKNs remain then the one associated with the higher priority station is selected. Otherwise a new PKN is randomly generated, PLK is reset, and PKA is zero, and AKA is as determined by the above procedure.

If the information held by the station has changed a KSP message transmission is scheduled.

Preventing data delay

MACsec can use the packet number (PN) field to prevent data replay, coincidentally discarding misordered frames. However this does not protect against the receipt of a whole sequence of frames delayed by an attacker. At the very least such an attack would likely disrupt the operation of network configuration protocols, in parts of the network beyond the immediate LAN. This is the very sort of behavior that MACsec is meant to protect against.

Each KSP participant maintains the age (LKA) of the latest (or last) key used for transmission, and the age (OKA) of the previously used (old) key if used for transmission within the last two seconds. Each KSP message contains the current age of those keys, according to the transmitter, and values for the nextPN (LPN, and OPN) for transmission with that key, recorded not less than ½ second and not more than 2 seconds prior to the KSP message transmission.

If the received age of either key is greater than that recorded by the station then the latter is updated by the received value. In this way the LKA and OKA increase at at least the speed of the fastest clock used by any of the stations.

Frames received by MACsec using the LKN are discarded if their PN values are lower than LPN, even if replay protection is disabled[25]. OPN is used to check frames received using the OKN.

Frequent transmission of KSP messages, preferably once every ½ second, is required to protect against data delay. Since the MI/MA tuples for all CA members may not fit in a single KSP message and it is undesirable to multiply the number of messages that are sent so frequently, the LKN and LKA are used as a proof of message liveness for reception of the LPN and OPN values.

Changing CAs and CAKs

While supporting continuous connectivity through changes of SAKs for a single CAK is a necessity for KSP, CAKs may also need to change with minimum disruption. Reasons to change a CAK include protecting against CAK compromise, using one CAK to bootstrap acquisition of another with stronger associated authorization, and excluding past CA members.

Each instance of KSP is associated with a single CAK value, but a single station can operate a number of KSP instances. Connectivity can be maintained as stations transition from one CA to the next[26], but at the same time the SAKs for different CAKs need to be quite distinct. Each station only transmit using one SAK at a time, and can thus be said to only be in one CA at a time, even if reception using different SAKs persists for a short period.

For a successful transition between CAs, MACsec frames should not be transmitted by a station using the new CA until that station and all others capable of using the new CAK are capable of both transmitting and receiving frames.

While a station may posses two CAKs and transition from using one to the other, the decision as to which is to be preferred is outside the scope of KSP. For a successful transition all stations that possess both CAKs need to have the same preference.

[25] Timeliness protection should be subject to management control independently of replay protection.

[26] Notably the MAC Service does not require MAC_Operational to transition false on a shared LAN just because connectivity to some stations has been lost (provided the remaining stations are fully connected).

KSPDU Format

Each KSPDU begins with a number of fixed format fields. These

- provide protocol discrimination and versioning
- identify the key and the initial value used to protect the PDU
- identify the transmitter, globally and uniquely by its SCI, and this particular powered-up instance of the transmitter, together with its age by its MI/MA
- advertise the PKN, PLK, PKA, and AKA fields to support selection of keys
- provide the key numbers and key ages for the two most recently used keys, together with the association numbers (ANs) that identify the associated SAs and lowest acceptable PNs for recently transmitted frames on those SAs

These fields are followed by list of MI/MA tuples that constitute a proof of liveness to the stations that use those MIs.

Finally the KSPDU may contain any number of octets of data transported on behalf of a protocol whose function is to agree new master keys.

The 128 bit identifiers used by the protocol are generated independently at random or using a method that is indistinguishable from random, as is the 96 bit IV. The 64 bit identifiers are either EUI-64, EUI-48+16, or EUI-48+12+4, i.e. globally unique and based on the same allocation procedures as MAC Addresses[27]. All timers are 64 bits, counting in milliseconds. The difference between 1024 milliseconds and one second is unimportant in the context of this protocol.

The details of the KSPDU fields follow. They are protected using GCM. All fields from and including the PCI to the end of the message are integrity protected by the trailing ICV, the fields from the MI through end of the message are confidentiality protected. The minimum size of a KSP message is 150 octets. If transmitted using directly on physical media, with 6 octet destination and source addresses and no other frame multiplexing or protocol discrimination components, this becomes 162 octets.

KSP Ethertype – 16 bits

PCI – PDU Control Info., 16 bits total

- **Version**, 4 bits
- **LAN**, Last/Latest AN, 2 bits
- **OAN**, Old AN, 2 bits
- **PLK**, 1 bit
- **MO**, 1 bit
- **Spare**, 6 bits

CKI – CA Key Identifier, 64 bits

IV – Initial Value, 96 bits

SCI – Secure Channel Identifier, 64 bits

MI – Member Identifier, 128 bits

MA – Member Age, 64 bits

PKN – Proposed Key Number, 128 bits

PKA – Proposed Key Age, 64 bits

AKA – Acceptable Key Age, 64 bits

LKN – Last/Latest Key Number, 128 bits

LKA – Last/Latest Key Age, 64 bits

LPN – Lowest acceptable PN for frames transmitted using the LKN, 32 bits

OKN – Old Key Number, 128 bits, used within the last two seconds (zero if unused)

OKA – Old Key Age, 64 bits

OPN – Lowest acceptable PN for frames transmitted using the OKN, 32 bits

Live List Length – number of entries in the following list of MI/MA tuples, 16 bits

MI/MA tuples – 128 + 64 bits each

User Data Length – number of octets following

User Data – octets

ICV – Integrity Check Value, 128 bits

KSP Addressing

On physical, as opposed to virtual, LAN media the destination MAC Address used with each KSPDU is the Bridge Group Address specified in IEEE Std 802.1D. Use of this address ensures that the KSP participants that exchange frames are attached to the same LAN. The source MAC Address should be an address associated with the transmitting station, it plays no particular role in KSP.

[27] It is however essential that there is no way provided to an administrator to change the lower 48 bits of these identifiers. There is absolutely no requirement that they be correspond to any MAC address currently in use.

KSP Transmission

KSP transmissions are rate limited and, if a station has received proof of liveness from more than one other station, subject to a small random jitter. Normally transmissions occur at roughly ½ second intervals, but the rate limiter allows a short burst, sufficient for connectivity to be established without any timer delay for a point-to-point CA.

Point-to-point examples

While supporting fault-tolerant multipoint connectivity, KSP works simply for two stations attached to a point-to-point link. There is no need for either station to remember past keys if this is the only scenario of interest to them. KSP provides the SAK with a three message exchange which can be originated by either party. A fourth message confirms receipt of the key by the originator's peer[28]. In the following scenarios the two stations are A and B, A having higher priority, and obvious, unimportant and null fields have been omitted.

If, following power up and initialization of the physical media, A happens to transmit just before B:

A installs K_A, B installs K_B

A → MI_A,MA_A,PKN=K_A,LKN=K_A →B

A← MI_B,MA_B,PKN=K_B,LKN=K_B,MI_A,MA_A ←B

A → MI_A,MA_A,PKN=K_A, LKN=K_A,MI_B,MA_B →B

B installs K_A, and knows A has

A← MI_B,MA_B,PKN=K_A, LKN=K_A,MI_A,MA_A ←B

A knows[29] B has installed K_A

Both A and B transmit and receive.

Assume that A's first transmission is lost, B having other power up initialization tasks to perform after A perceives the physical LAN media as being ready. Then the sequence of messages proceeds as follows:

A← MI_B,MA_B,PKN=K_B,LKN=K_B ←B

A → MI_A,MA_A,PKN=K_A, LKN=K_A,MI_B,MA_B →B

B installs K_A, and knows A has

A← MI_B,MA_B,PKN=K_A, LKN=K_A,MI_A,MA_A ←B

A knows B has installed K_A

Both A and B transmit and receive.

Only three messages are required, since B didn't change A's choice of K_A as the key. The above sequences assume rapid installation of the keys, if this took some time the protocol would follow the pattern above but additional messages might be required to report changes in LKN before transmission could begin.

If the point-to-point LAN service is temporarily interrupted by some physical fault, then when service is restored both A and B transmit:

A → MI_A,MA_A,PKN=K_A, LKN=K_A,MI_B,MA_B →B

A ← MI_B,MA_B,PKN=K_A, LKN=K_A,MI_A,MA_A ←B

and each can start communicating again, in the knowledge that the service interruption wasn't caused by rearrangement of the physical cabling.

However if the physical interruption was caused by A's connection now being plugged into C, then the subsequent sequence of messages will follow the pattern of one of our first two examples.

If A and B are communicating using K_A and B gets close to running out of PN space, it can propose a new key, K_C say, as follows (MI/MA tuples up to date and known to both A and B):

A ← PKN= K_C,LKN=K_A ←B

A installs K_C

A → PKN=K_C, LKN= K_C,OKN=K_A →B

B starts transmitting using K_C

A ← PKN=K_C, LKN= K_C,OKN=K_A ←B

A starts transmitting using K_C

Multipoint examples

Assume A and B are already communicating using a shared media LAN, when C is attached. As before the precise sequence of messages depends on relative timing at the two stations. For example, does C receive a message from A or B before transmitting its own? The following is a typical message sequence (in this sequence MI/MA tuples are simply denoted by M) on the optimistic assumption that the system of which C is a part has been powered up for longer than K_A has been is use, so C starts with an AKA that is longer than the age of K_A .

C installs K_C

C → M_C, PKN= K_C,LKN= K_C

A → M_A, PKN=K_A, LKN=K_A, M_B, M_C

C installs K_A, and starts transmitting.

If C had arrived in the middle of a key change, and received a message from a station which proved itself live, had neither K_A as the LKN or the OKN, and had a higher priority than A, then C would transition MAC_Operational false again.

[28] The KSP exchange thus provides the equivalent of the customary 4-way handshake.

[29] Arguably this step could be avoided by A simply receiving data from B, since B can transmit and receive following the third message. KSP will send this fourth message though, it makes its operation independent of the data frames.

If C had just powered up and had no memory of a previous key, and consequently an AKA of zero, then the sequence might proceed as follows:

C installs K_C

C → M_C, PKN= K_C,LKN= K_C

A → M_A, PKN=K_A, LKN=K_A, M_B, M_C

C → M_C, PKN= K_C,LKN= K_C, M_B[30], M_A

B receives and installs K_C

B → M_B, PKN=K_C, LKN= K_C,OKN=K_A, M_B, M_C

B will not use K_C until it hears from A

A receives from C, installs K_C

A receives from B, transmits using K_C

A → M_A, PKN=K_C, LKN=K_C, OKN=K_A, M_B, M_C

B, C receive and transmit using K_C

Connectivity between A and B is maintained throughout the process of changing from K_A to K_C. If A fails before transmitting its final message, B will still move to using K_C, but there will be a delay of two seconds after which it will be assumed that A is no longer live, so the new key can be used without A's approval.

If A recovers within a few seconds, before the key is changed again then the following sequence is typical:

A → M_A, PKN=K_C, LKN=K_C

B → M_B, PKN=K_C, LKN= K_C,OKN=K_A, M_B, M_C

A starts transmitting and receiving again.

[30] Note C acquires M_B from the message from A. This implies no judgment on the part of C that station B is actually live.