# Security considerations and proposal for MACsec key establishment

Brian Weis
5/15/06

This document describes security considerations for MACsec automated key establishment protocols. It also takes makes claims regarding the effective security that is possible for MACsec automated key establishment given the exclusive use of secret key cryptographic algorithms. Finally, it proposes one possible method of key establishment that takes into account the security considerations, as well as the cost of mechanisms that can provide the possible effective security.

## 1. Introduction

The IEEE P802.1AE working group has created MACsec [5], a link level security standard for 802 LANs an MANs. MACsec does not directly address how keys are obtained for encryption, although it does include a management interface for requesting and obtaining keys from key establishment protocols for Connectivity Associations (CAs).

The IEEE P802.1af working group was formed to address the needs of device authentication and data encryption key generation. Current plans of the working group require a device to obtain a Connectivity Association Key (CAK), which is a long-term master secret key. Proof of possession of the CAK is suitable for proof that it has been authenticated using an IEEE 802.1X framework, and is authorized to participate on a particular LAN.

A key establishment protocol is required to generate one or more Secure Association Keys (SAKs), which are the secret keys that IEEE 802.1AE uses to encrypt data packets on the LAN. Some 802.1 LANs are shared media LANs with multiple stations, so SAKs are assumed to be group keys shared between two or more stations on the LAN. Therefore, the IEEE P802.1af key establishment protocol must be a group key management protocol.

This document investigates the security ramifications of the IEEE P802.1af system design, and in particular the possible resulting security properties of a group key establishment protocol fitting into this system design. It also proposes one possible group key establishment protocol that meets the requirements of the P802.1af system design.

## 2. Overview

The IEEE P802.1 working group defines standards for 802 LAN and MAN bridges. The types of network topologies that deploy 802 networks are varied, and include at least the following topologies.

- One or more end stations (i.e., PC's, network attached printers, IP telephones) connected to bridges
- Interconnected bridge ports
- Provider bridged network

IEEE 801.1AE group standardizes an encryption method for encrypting packets within each of these scenarios.

### 2.1. Terminology

**Advanced Encryption Standard (AES)** – a FIPS-approved symmetric block cipher cryptographic algorithm that can be used to protect electronic data. [7]

**Association Number (AN)** – A number that is concatenated with the Secure Channel Identifier to identify a Secure Association. [5]

**Connectivity Association** (CA) – A security relationship, established and maintained by key agreement protocols, that comprises a fully connected subset of the service access points in stations attached to a single LAN to be supported by MACsec. [5]

**Connectivity Association key (CAK)** – The long term key associated with a Connectivity Association. [14]

**Connectivity Association Key Identifier (CKI)** – An identifier for a particular CAK. [14]

**Key establishment** – A function in the lifecycle of keying material; the process by which cryptographic keys are securely established

among cryptographic modules using manual transport methods (e.g., key loaders), automated methods (e.g., key transport and/or key agreement protocols), or a combination of automated and manual methods (consists of key transport plus key agreement). [1]

**Key wrapping** – A method of encrypting keys (along with associated integrity information) that provides both confidentiality and integrity protection using a symmetric key. [1]

**Random Number Generator (RNG)** – An algorithm or method used for cryptographic applications that typically produces a sequence of zero and one bits that may be combined into sub-sequences or blocks of random numbers. [9]

**Secure Association Keys (SAK)** – The secret key used by an SA. [5]

**Secure Channel (SC)** – A security relationship used to provide security guarantees for frames transmitted from one member of a CA to the others. An SC is supported by a sequence of SAs thus allowing the periodic use of fresh keys without terminating the relationship.[5]

**Secure Association Key Identifier (SKI)** – An identifier for a particular SAK.

## 2.2. Acronyms

**AES** – Advanced Encryption Standard

**AN** – Association Number

**CA** – Connectivity Association

**CAK** - Connectivity Association Key

**CKI** – Connectivity Association Key Identifier

**RNG** – Random Number Generator

**SAK** – Secure Association Key

**SC** – Secure Channel

**SKI** – Secure Association Key Identifier

## 2.3. Operational Considerations

### 2.3.1. Constraints

There are many operational constraints on protocols and software operating on a NIC card or Ethernet bridge. These constraints may not be familiar to cryptographic protocol designers who typically deal with devices with more capabilities.

1. The CPU should not be expected to be powerful enough to perform extended mathematical operations, including public key cryptography. Thus, a key establishment protocol should use only secret key technology for device authentication, packet authentication, and packet encryption.[1]
2. There is typically little or no possibility of user configuration of Ethernet security. E.g., timer durations are not likely to be configurable.
3. Devices are assumed to have little or no persistent storage at the NIC or bridge port level. Long-term keys derived from an IEEE 802.1X exchange and other persistent configuration values are the responsibility of higher levels of the system (e.g., perhaps derived as part of the IEEE 802.1X exchange).

### 2.3.2. Resources

We assume that a key establishment protocol has a timer service available to it. These timers interrupt the software at a specified future time, identified by a time interval.

# 3. MACsec Key Establishment Security Review

## 3.1. Overview of Security Properties

This section described the security properties common to key management protocols, including commentary on whether those security properties can be achieved in this environment given the operational constraints.

### 3.1.1. Confidentiality

Ensuring confidentiality of key establishment packets using a key derived from the CAK is possible, but may not be required if no keying material is carried in the packet. If the key establishment packets are passing SAKs or other sensitive information, at least that portion of the packet must be encrypted for confidentiality.

[1] Other parts of the system may use public key cryptography (e.g., for device authentication to an authenticator). But the key establishment protocol between stations on a LAN should not expect to have the resources necessary to use public key cryptography.

### 3.1.2. Data Integrity

Ensuring that an unauthorized station has not modified packets is possible using a key derived from the CAK.

### 3.1.3. Source Origin Authentication

Ensuring that the origin of the packet is the true origin of the packet is called source origin authentication. Source origin authentication requiring public key cryptography (e.g., digital signatures) is best, but not possible for the present application due to CPU constraints. Other forms of source origin authentication using only secret keys are available in the literature [12],[11] but they require both loose time synchronization between stations and a delayed exposure of keys. Neither of these methods are acceptable to the present application. Therefore, no source origin authentication may be possible.

### 3.1.4. Anti-replay Protection

Ensuring that a key establishment packet previously sent on the LAN is not accepted a second time is an important property. An attacker replaying old packets could cause an older key to be re-used, for example.

### 3.1.5. Liveness

Ensuring that a key establishment packet has been recently sent (i.e., is a live packet) is an important property. An attacker delaying the propagation of a packet could cause it to be accepted much later and confuse the state of a key establishment system.

### 3.1.6. Denial of Service Protection

An attacker can always flood the LAN with replayed or invalid packets. A key establishment protocol must be able to quickly identify these packets and drop them without undue processing. If an attacker succeeds in flooding the LAN such that all communication is disrupted, the key establishment protocol must recover in a timely fashion at the conclusion of the attack and allow encrypted data traffic to begin immediately.

### 3.1.7. Forward and Backward Secrecy

Backward secrecy ensures that when a new station member joins the LAN that it cannot learn about the previous group keys. Similarly, forward secrecy ensures that when a station leaves a LAN that it cannot compute future group keys. These properties cannot be maintained by an IEEE 802.1AE key establishment protocol, because keys must be either derived from or protected by the CAK. The CAK is a long-term key that does not change when stations join or leave the LAN.

### 3.1.8. Byzantine Attack Protection

A “Byzantine Attack” occurs when an authorized member of the LAN intentionally or accidentally disrupts the communications of the group. When all trust is based on holding a long term secret key (i.e., the CAK), and when source origin authentication is not maintained, it is not possible to protect against these attacks. It will be possible for any station holding the CAK to disrupt the key establishment protocol at any time and for any duration. Therefore, protection against Byzantine Attacks is not possible.

## 3.2. NIST Key Management Guidelines

The United States National Institute of Standards and Technologies (NIST) develops computer security standards and guidelines [10]. These standards and guidelines are considered relevant internationally.

NIST has produced the FIPS 140-2 cryptographic standard to which many organizations requisitioning cryptographic systems require adherence. Thus, IEEE 802.1AE and associated key establishment protocols should be designed to comply with FIPS 140-2.

NIST has also published recommendations for cryptographic key management [1], [2]. These recommendations are in part intended to aid cryptographic protocol designers develop new cryptographic protocols. As such, any new cryptographic method of establishing or selecting keys will be measured against these recommendations during a FIPS 140-2 evaluation.

Given the previously described severe constraints, it may not be possible for those recommendations to be followed completely. In that case, rationale for deviation must be maintained. However, those recommendations should be followed whenever possible.

## 4. LAN-based Key Server Protocol

The LAN-based Key Server (LKS) protocol is a proposed method of providing keys to IEEE 802.1AE stations. This method uses a traditional group key management paradigm where one station on the LAN acts as a key server that derives and distributes the current SAK to other members of the same Connectivity Association.

Pre-determining a single LAN-based key server is not reliable. Rather, the stations on the LAN first elect one of themselves to be the key server. Typically, a re-election does not occur until either the elected key server becomes non-responsive, or any of the stations becomes uncertain as to the identity of the elected key server. The latter case can happen if two different stations each forward a message defining different SAKs.

Although a single station is responsible for deriving SAKs, any station may request that a new SAK be generated. This may be because a station believes that the key lifetime is about to expired, or may be because its AES-GCM IV space is about to be fully used. In addition, the elected key server may choose to replace the current SAK based on other events. For example, if the elected key server observes a new identity on the LAN it must create a new key to guard against a rebooted station re-using the same AES-GCM IVs with the same key.

The LAN-based key server chooses SAKs randomly. Assuming keys are generated uniformly over the number space, the "Birthday Paradox" tells us that on average a 128-bit AES key will not be repeated until a set of $2^{64}$ keys have been generated. As such, there is no need to keep track of previously generated SAKs.

LKS incorporates a number of design elements from the Key Selection Protocol [14], another proposed key establishment protocol for IEEE 802.1AE. In particular, some KSP terminology is maintained for ease of comparison. The KSP anti-replay and "liveness" protection mechanisms appear to be the most efficient mechanisms for this application, and this has been incorporated as well.

## 4.1. Goals

The following statements describe goals of this protocol.

1. Reliability is of the utmost concern. An 802 LAN should not be left without a current SAK except under exceptional circumstances (e.g., a persistent DoS attack).
2. Provide secure connectivity within the first second of the underlying LAN service becoming available.
3. Adapt quickly to the addition of new stations on the LAN without disrupting connectivity between existing stations on the LAN.
4. 802 LANs can be either point-to-point (e.g., between bridge ports) or a shared media LAN with multiple stations (e.g., using a repeater). A single key establishment protocol for a LAN should support both configurations, without pre-selection of one or the other.
5. Operate without requiring pair-wise communication between all stations.

## 4.2. Design Considerations

In addition to the previously described operational constraints, the following design considerations were maintained for the development of LKS,

- SAKs are transferred between stations using guidance from Section 4.2.5 of [1].

## 4.3. Cryptographic Operations

Stations supporting this protocol must have the following capabilities:

- AES protocol supporting 128 bit keys, and the following modes of operation: Electronic Code Book (ECB), and Cipher-based Message Authentication Code (CMAC) [4].
- Strong random number generator (RNG). If a non-deterministic RNG (e.g., hardware RNG) is not available, then sufficient entropy must be available to create a good quality seed for a deterministic RNG.[2]

This section summarizes the cryptographic operations of this protocol.

### 4.3.1. SAK Generation

SAKs are generated using a strong RNG, preferably one approved by FIPS 140-2, listed in its Annex C [8].

The SKI identifying a SAK is also generated randomly. Although not cryptographically necessary, use of a cryptographically strong RNG is recommended,

### 4.3.2. Deriving Keys from the CAK

As previously described, the only long-term shared secret available between stations is the CAK. This secret must be used for two purposes: to encrypt SAKs as they are distributed between stations, and provide an integrity check on the LKS messages. In

[2] A deterministic RNG can often be implemented in software. Several are identified in the FIPS 140-2 Annex C [8].

order to use the CAK for these two purposes, a key hierarchy rooted to the CAK is defined.

#### 4.3.2.1. SAK Distribution

SAKs are distributed from a station generating the key to other stations on the LAN. The keys must be encrypted during transit so that only authorized stations (i.e., those holding the CAK) are able to recover the key.

Keys are encrypted using a Key Encrypting Key (KEK), which is a sub-key derived from the CAK as follows:

KEK = AES-ECB(CAK, 0x0)

CAK is a 128-bit AES key, and the encrypted data is a single 128-bit block with the value '0x0'. The KEK is given as input to a key wrapping algorithm to protect the SAK between stations. The default algorithm for protecting the SAK is the AES Key Wrap [13].[3] The AES Key Wrap default IV (defined in [13]) MUST be used.

#### 4.3.2.2. Message Authentication

Message authentication is achieved by including an Integrity Check Value (ICV) in each message. The ICV is computed as a cryptographic operation over the bytes of the message with a secret key. All bytes following the IEEE 802.1 header in the message are included in the ICV generation, excepting the ICV field itself.

The key used in the ICV generation is called the ICV_KEY. The ICV_KEY is a derived from the CAK as follows:

ICV_KEY = AES-ECB(CAK, 0x1)

where CAK is the AES key, and the encrypted data is a single 128-bit block with the value '0x1'.

The default algorithm generating the ICV is CMAC [4] using an AES-128 key. The output of the ICV is a 128 bit value, computed as follows:

ICV = AES-CMAC(ICV_KEY,M,128)

and M is defined as the protocol message bytes to be authenticated.

[3] This specification has also been adopted by IEEE P802.16e/D9 for purposes of distributing session keys.

### 4.4. Anti-replay, Liveness, and Denial of Service Protections

It is critical that a key establishment protocol be able to differentiate between packets that have never been seen before and older "replayed" packets. The protocol should never accept the same packet twice as an original packet. A means of anti-replay protection is required in order to make this determination.

Similarly, a key establishment protocol should be able to tell the difference between a packet that was recently sent, and one that was not recently sent. A receiver should be able to make this delineation even if the receiver has never previously seen the delayed packet (i.e., the packet is not a replayed packet). A means for checking "liveness" of the packet is required.

A common anti-replay protection method is for a sender to maintain a counter, which is incremented for each packet sent. When each packet sent has a unique monotonically increasing counter value, it is called a sequence number. Each station must maintain its own sequence value, and each receiver must keep track of the most recent sequence number seen from other stations.[4] This mechanism partially satisfies the anti-replay protection need of this protocol. To be consistent with sequence number terminology in KSP, a sequence number is called a Message Number (MN) in this paper.

Some properties with sequence numbers are:

- When used over time sequence numbers will eventually reach the largest possible value and wrap back to zero. Since a wrapped sequence number appears to be a replayed packet to receivers, receivers need to be notified before this event occurs.
- Receivers typically join a group at different times. When they begin to participate they will not know a priori what sequence numbers for sending stations are valid. Care must be taken they are not fooled into accepting replayed packets, especially in the face of an attacker replaying packets with high values of sequence numbers such that valid packets from the sender appear to be very old replayed packets.

[4] It is common for security systems to maintain a "window" of previously received sequence numbers in addition to the most recently received sequence number. This would be of little use to LKS because any message other than the most recent message sent from a peer is considered stale.

To mitigate these issues, each station randomly chooses an identity that is associated with a particular set of sequence numbers. When the number space is about to expire, they choose a new identity before beginning the sequence number space at its lowest value. To be consistent with sequence number terminology in KSP, a station identity is called a Member Identifier (MI).[5]

Although it is an unlikely event, two stations could choose the same MI value. If a station detects that another station is using its MI value, it must immediately change its MI value.

Denial of service attacks that replicate packets will be stopped by the anti-replay measures described above. Attacks that flood a link with packets cannot be stopped, but once the flooding terminates the anti-replay measures will quickly take affect again and the protocol will protect itself from replayed and invalid packets.

### 4.4.1. Determining Anti-replay and Liveness

In order to determine anti-replay and liveness, each station maintains the following state:

- A MI value and current MN representing the station's own the most recently used MN. The MN begins at 1 and is monotonically incremented for each packet that it sends. When the MN value is in danger of wrapping a new MI value is chosen. This method guarantees that a station's MI/MN pair is always unique.
- A list of MI/MN pairs from stations that have been proven to be live, called the Live Peers List. A station adds a peer station is to its Live Peers list when the peer reflects the current MI/MN pair of the receiving station in its message. This reflection of an MI/MN pair proves that the peer recently received a packet containing the station's MI/MN pair.
- A list of MI/MN pairs from entities from which a packet was received, but that peer has not yet been proven to be live. That is, the most recent message from that peer did not include the current MI/MN pair of the receiving station. This list is called the Potential Peers List.

In summary, a station maintains an MI/MN pair used when sending its own packets. It also maintains an MI/MN pair documenting the most recent stations that have recently sent messages, and an MI/MN documenting stations that appear to be sending messages but have not been proven to be live.

When a station receives a packet from a peer, it makes a determination as to the reliability of the peer. The following logic is followed, and is also shown pictorially in Figure 1.

1. The receiving station checks its locally stored Live Peer List and Potential Peer Lists for the sender's MI value.
2. If the sender's MI is found in either list, the MN received in the message is compared to the MN stored on the list.
   a. If the MN in the packet is equal to or smaller than the stored MN, the packet is considered to be a replayed or delayed packet and it is dropped.
   b. If the receiver's MI/MN pair is found in either Peer List of the message, the sender is considered to be a LIVE PEER (because they claim to have seen a recent packet from the receiver). If the sender's MI/MN pair is found in the Potential Peer list, the peer's MI/MN pair is also moved to the Live Peer List.
   c. If the receiver's MI/MN pair is not found in either Peer List of the message it is added to the Potential Peer list and considered to be a POTENTIAL PEER.
3. If the sender's MI is not found in either of the locally stored Live Peer List or Potential Peer List, the sender is considered to be a POTENTIAL PEER, and is added to the Potential Peer List.

[5] The station MAC address is not used as an identity, because it cannot be changed when the MN needs to be wrapped.

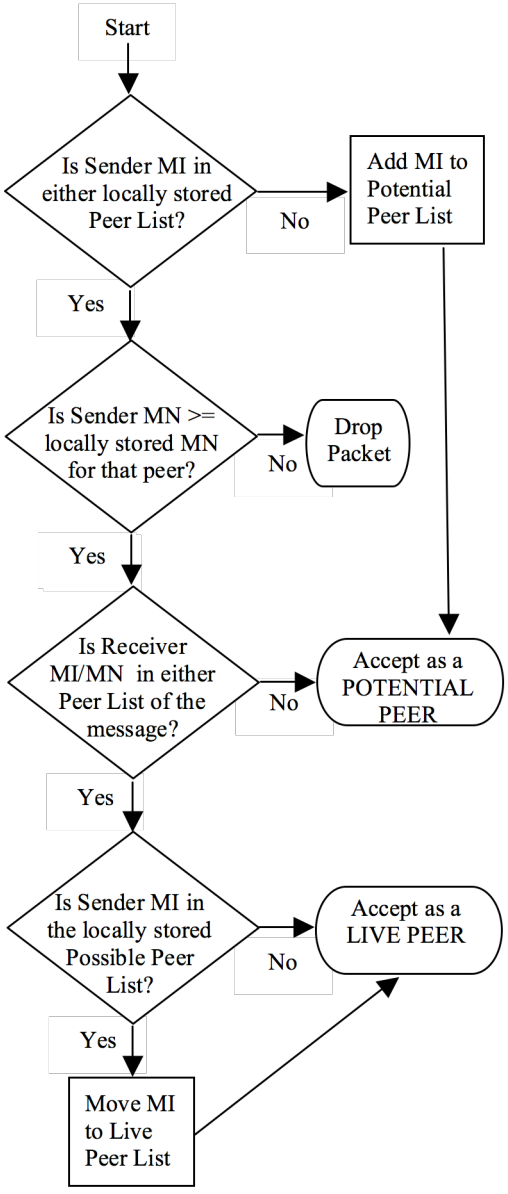



**Figure 1. Anti-Replay/Liveness Receiver Processing**

Each entry in the Live Peer and Potential Peer list also documents when the message containing the MI/MN pair was received. The station periodically reaps entries that are older than a given threshold. This mechanism allows a station to detect when other stations have become non-responsive.

## 4.5. Key Server Election Process

LKS employs the key server model of group key distribution. However, in most cases a single pre-ordained key server would be useless, since its absence at any time would result in the rest of the devices on the link being unable to communicate.[6] Thus, LKS includes a key server election process. As long as one station is active, the LAN will have a key server.

For efficiency, key server election is not a separate protocol exchange. Each LKS message contains enough state for each station to determine at any given time which station should be the key server on the LAN. The "election" process consists of an evaluation of peer state and the station's own state. This evaluation process happens as the result of any of the following events:

- A station determines that no station is currently acting as key server. (This includes the case where no live peers are found.)
- A station deems the current key server to be non-responsive.
- A message is received from a station claiming to be a key server, and this station is not the current key server.

It is possible for two or more stations to believe they are the key server for the LAN. This will happen when a new station comes online, or a partitioned network is joined. In this case, each station applies an election heuristic to determine which station should retain the role of Primary.

During an election, each station numerically compares the Member Identifiers (MIs) of all stations claiming to be Primary. The station retaining the Primary Role is the station with the highest MI value.[7] If two devices claiming to be Primary also choose the same MI value, then the device with the highest Secure Channel Identifier (SCI) value is chosen as the key server.[8]

---

[6] The exception would be when hosts use a key agreement protocol to join a network via a network access point, but don't expect to communicate amongst themselves. The absence of a key server would then be no worse than the absence of the network access point itself.

[7] Use of the MI provides the following advantage: Some devices (e.g., switch ports) are natural preferred key servers in some use cases. Those devices could skew its choice of MI values to be in the high end of the MI namespace.

[8] If a station believes that two devices have the same SCI and MI values, then it should log an error, since the odds of this accidentally happening should be quite low.

## 4.6. State Machine Overview

Stations begin after obtaining the CAK. The state machine begins in the Initialize state where it initializes state variables. It then transitions to the waiting state and reacts to events as they happen. Figure 2 shows an overview of the LKS state machine.

### 4.6.1. Protocol State Overview

Each station running this protocol maintains the following state:

- Current CAK and its identifier (CKI).
- The station's own identity (MI, SCI), as well as the most recent sequence number used (MN).
- The station identity currently believed to be in the P role
- The role of the current station.
- The current SAK (and its identifier (SKI))
- The identities of stations known to be live on the LAN, the most recent sequence number that has been seen in a valid packet sent from that station, and whether the station claims to take the Primary role.
- The identities of stations that appear to have recently sent a packet, the most recent sequence number that has been seen in a valid packet sent from that station, and whether the station claims to take the Primary role.

### 4.6.2. Protocol Messages and Timers

LKS supports a single message type, called a SAK_MSG. A station acting in the Primary role (called P) broadcasts the current SAK to other stations using a SAK_MSG. A station not in the P role uses the SAK_MSG as a heartbeat message declaring their liveness to other stations.

The following timers are required for this protocol:

- Heartbeat Timer, to cause a station to send a message periodically. A station in the P role during the SAK transmission detects when a SAK is will expire before the next scheduled heartbeat and creates a new SAK.
- Management Timer, to periodically purge state peer entries in the Live Peer and Potential Peer lists.

### 4.6.3. Implementation Configuration Values

The following values need to either be configured, set as an implementation default, or defined in this document. (TBD)

- Maximum lifetime of a SAK
- Length of the period between SAK_MSG transmissions.

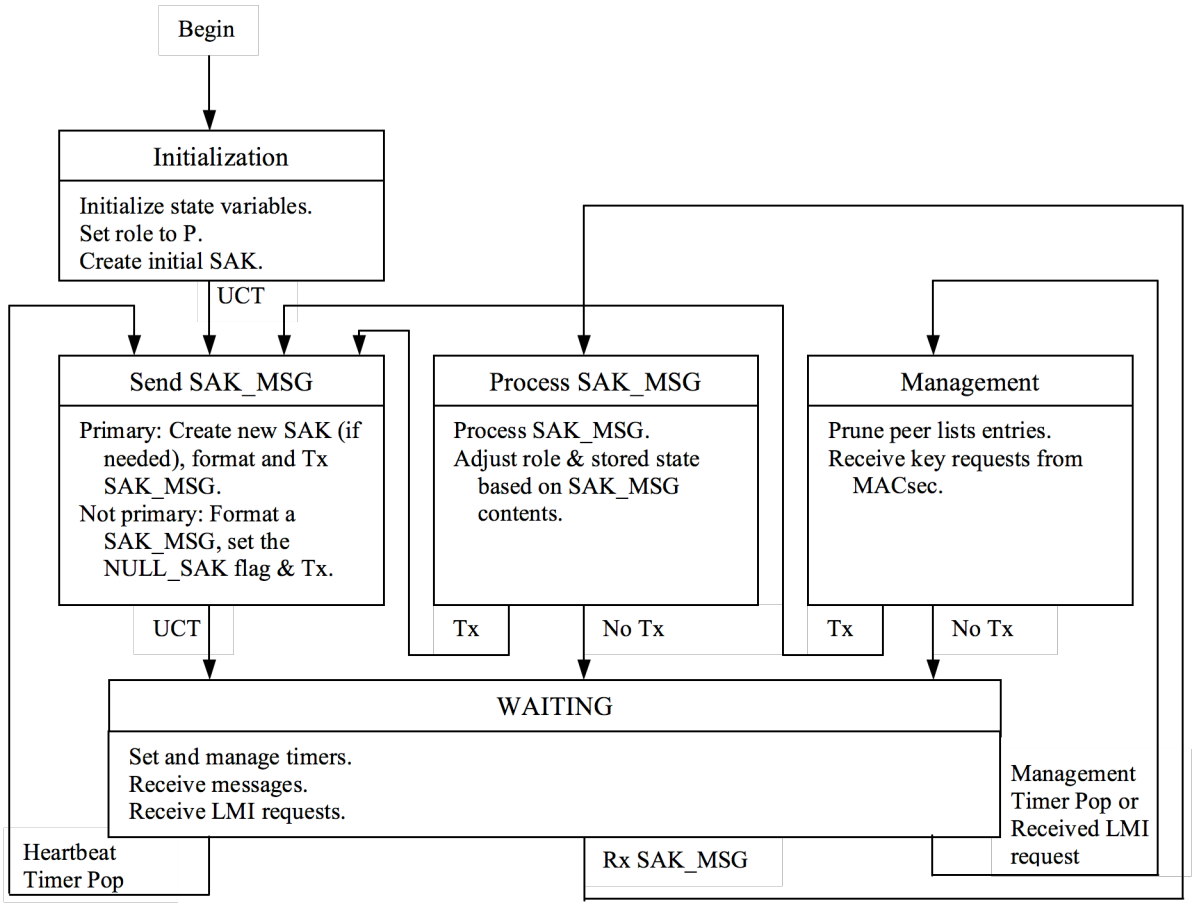



**Figure 2. LKS State Machine**

## 4.7. Initialization State

This state is the beginning of the key agreement protocol, and is entered when the station obtains the CAK. Various state variables are initialized as follows.

- An MI value is randomly chosen.
- The MN is set to 0.
- The station's role is set to Primary (P)
- A SAK is randomly chosen.
- A SKI is randomly chosen.

When initialization is complete, the station unconditionally transitions (labeled "UCT") to the Transmit State to transmit its new SAK. The Initialization state is never re-entered unless an external event causes the state machine to start over.

## 4.8. Send SAK_MSG State

This state is entered whenever a SAK_MSG must be broadcast. Reasons for entering the Transmit State are:

- The station transitioned from the Initialization state,
- A Heartbeat timer popped, or
- The Process SAK_MSG state resulted in needing to broadcast a message in return.

A station in the primary role formats and transmits a complete SAK_MSG including the current SAK. For state machine simplicity, the decision to create a replacement for an expiring SAK is made in this state as part of formatting the message.

A station not in the primary role always adds the NULL_SAK flag when formatting the message and omits sending a SAK. The setting of the NULL_SAK flag also indicates to the receiver that they are not in the primary role.

Detailed actions performed in this state are described in Section 4.13.1.

## 4.9. Process SAK_MSG State

This state is entered when a SAK_MSG is received. The message is first authenticated and proven to be from a live peer. In all cases a receiver updates its peer state based on the liveness state in the message. Processing the message differs whether or not the sender of the message is in the P role.

The receiver is known to be not in the P role if it included the NULL_SAK flag set. If the receiver is also not in the P role, it has no need to further process the packet. But a receiving station is in the P role makes two additional checks, both of which result in the creation and transmission of a new SAK:

- The CREATE_NEW flag is set in the packet
- The sender has an MI value not previously stored in the Live Peer list.

If the message contains a SAK (i.e., the NULL_SAK flag is not set), and if the sender is currently believed by the receiver to be the group Primary, then it accepts and installs the new SAK. The only exception to this rule is if the receiver detects that the sender has not yet marked the receiver as a Live Peer in the message. In this case, the receiver cannot yet install the SAK because doing so may violate an AES-GCM security condition.

If the message contains a SAK from a sender NOT currently belived by the receiver to be the group Primary, then it performs the election process (described in Section 4.5).

Detailed actions performs in this state are described in Section 4.13.2.

## 4.10. Management State

Periodic and miscellaneous management events are handled in the Management state. Periodic background processing happens as a result of the Management Timer pop:

- Non-responsive ("dead") peers are pruned from the peer lists.
- After the peer lists have been pruned, the stations detects if the current Group P station has become non-responsive and performs the election process as defined in Section 4.13.3.

When LKS receives a request from LMI, it handles it in this state (TBD).

If a management event requires a new SAK (e.g., MACsec indicates that a new group key is necessary because its Packet Number is about to wrap), the station transitions to the Send SAK_MSG state. Otherwise, control returns to the Waiting State.

## 4.11. Waiting State

This state is entered when other states have completed. It waits until one of the following events effects a transition:

- A timer interrupts the software, indicating that some time interval has finished.
- A message is received from a peer.
- A Layer Management Interface (LMI) request is received from MACsec.

When the Heartbeat timer pops, control transitions to the Send SAK_MSG state to send a message. Every station does this, regardless of their current role. This periodic sending of packets proves to other stations that the sender is live.

When either the Management timer pops, control transitions to the Management state. When an LMI request is received, control is also transitioned to the Management state to handle the request.

When a SAK_MSG is received, control transitions to the Process SAK_MSG state to validate and store the new state in the message.

## 4.12. Frame formats

The following figure shows the general frame format for protocol messages. [NOTE: Frame formats are not complete. E.g.., little thought has yet been put into word alignment of fields.]

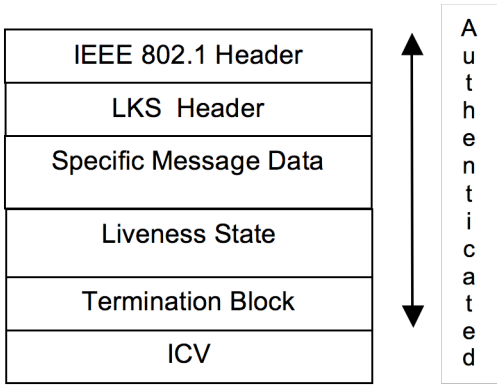



**Figure 3. PDU Format**

### 4.12.1. IEEE 802.1 Header

The header comprises three fields:

- **Destination Address (12 bytes).** A multicast address, confined by bridges to a single LAN.
- **Source Address (12 bytes)**. The station's own address.
- **Ethertype (2 bytes)**. A new Ethertype value defined for this protocol.

### 4.12.2. LKS Header

The LKS protocol header has the following fields:

- **Connectivity Key Identifier (8 bytes)**. The CKI identifies a particular Connectivity Association Key.
- **Secure Channel Identifier (8 bytes)**. The SCI is a MACsec identifier comprised of the station's MAC address and Port ID. The default MACsec Default Cipher Suite (the GCM-AES-128 Suite) depends upon all stations in the CA having unique SCI values. It is passed in the LKS header so that LKS can detect SCI collisions.
- **Member ID (4 bytes)**. This field contains the current identity for the sending station.
- **Message Number (4 bytes)**. This field contains sequence number that increases monotonically for each message sent by this Member ID. Receiver processing of this messages number provides anti-replay protection.

### 4.12.3. Specific Message Data

Each message contains the following fields.

- **Message Type (1 byte)**. This field defines the type of the message.[9]

| *Type* | *Value* |
|---|---|
| SAK_MSG | 0x1 |

  - SAK_MSG. All stations send this message type. A station in the P role includes a SAK to be used by the group. A SAK is sent without any policy describing how that key should be used (e.g., which Cipher Suite should be used). It is assumed that this policy is provided to the station via a management interface [802.1AE, Section 10.7] A station not in the P role includes the NULL_SAK flag (see below) but no SAK.

- **Flags (3 byte)**.

| *Type* | *Value* |
|---|---|
| CREATE_NEW | 0x1 |
| NULL_SAK | 0x2 |

  - CREATE_NEW. This flag is included when a station requires a new key (e.g., if its IV space is about to run out).
  - NULL_SAK. This flag indicates that no SAK is included in this message. When NULL_SAK is set, the remaining fields in the Specific Message Data are omitted.

- **Key Wrapped SAK Type (1 byte)**. Identifies the type of key wrapping around the SAK.

| Type | Value | Length of Wrapped Key |
|---|---|---|
| AES_KEY_WRAP | 0x1 | 16 bytes |

- **Key Wrapped SAK. (variable).** The SAK.
- **Secure Association Key Identifier (SKI) (4 bytes).** Identifies the current SAK.
- **Key Lifetime (4 bytes)**. The maximum duration (in seconds) that the station should expect to use this key. The actual lifetime of the key may be much smaller depending on other events that may cause a rekey before that time.
- **Association Number (1 byte)**. The AN in the CA in which this SAK should be installed, as well as attributes of the AN.

| AN value | Rx | Tx |
|---|---|---|
| 2 bits | 1 bit | 1 bit |

  - **AN Value**. The primary should alternate between the two values 0x01 and 0x10 for successive SAKs that it generates.
  - **Rx**. If set to 0x1 directs the station to receive on this AN.
  - **Tx**. ]If set to 0x1 directs the station to send on the AN.

  For a group key, both the Rx and Tx bits will be set.[10]

### 4.12.4. Liveness State

A station maintains liveness state for all peers on the LAN that are able to construct a valid ICV. The liveness state consists of MI:MN values for each peer. Peers that have recently sent a packet with the station's own NI:MN values are considered "live

[9] One message type has been currently defined, but this field provides extensibility.

[10] If the SAK were per-sender it would only have the Rx bit set.

peers"; peers that have not are considered "potential peers".

The following structure comprises the liveness state.

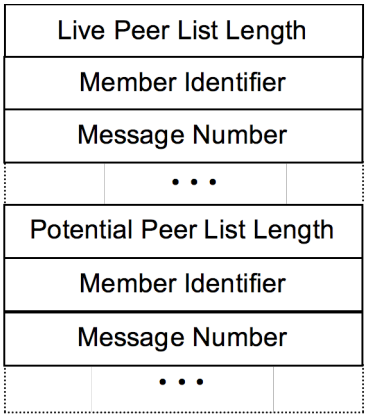



**Figure 4. Liveness State**

- **Live Peer List Length (2 bytes)**. Number of MI/MN value pairs for peers that have recently shown to be live.
- **Member Identifier (4 bytes)**. Identifier of an authenticated station on the network.
- **Message Number (4 bytes)**. Latest sequence number observed for the Member ID.
- **Potential Peer List Length (2 bytes)**. Number of MI/MN value pairs for peers that have apparently recently sent a packet, but that have not yet reflected back a recent MI/MN pair for the sending station.

### 4.12.5. Termination Block

This field consists of two bytes, and has the value 0x0. It marks the end of message data fields. Future versions of this protocol may add TLVs with a two byte "type" field which are non-zero.

### 4.12.6. ICV

The ICV is calculated as described in Section 4.3.2.2 (16 bytes).

## 4.13. Message Operations

Sending or receiving a message will have nearly identical semantics, with the exception dealing with the specific message data as described above.

### 4.13.1. Transmitting Messages

The following steps are followed to create and transmit a message.

1. Create the IEEE 802.1 header, populating the header fields as defined in Section 4.12.1.
2. Add the LKS Header, populating the header fields as defined in Section 4.12.2.
3. Add a SAK according to the following rules.

```
IF the station is in the Primary role
    IF SAK lifetime is near its end
        Create new SAK, its identifier, and
            lifetime.
        Add new SAK to message.
    ELSE
        Include current SAK in message.
ELSE
    Set the NULL_SAK flag in the message
        indicating both that there is no SAK
        and the sender is not in the P state.
    IF the prior state called for a new SAK
        Set the CREATE_NEW flag in the
            message.
```

4. Add Liveness State, as defined in Section 4.12.4. The Live Peer List is populated from the locally stored live peer list described in Section 0. Similarly, the Potential Peer List is populated from the locally stored live peer list.
5. Add the Termination Block, as defined in Section 4.12.5.
6. Create the ICV as defined in Section 4.3.2.2, and add it to the message.

### 4.13.2. Receiving Messages

The following steps are followed when receiving a message with the Ethertype defined for this protocol, and with the correct multicast destination address. In all cases, an error condition results in processing of the packet being aborted. Errors should be logged, and should be rate limited.

1. Verify liveness & freshness of the packet. Doing this before the ICV check mitigates some DoS attacks.
   a. If either peer list contains the receiving station's MI, verify that the station's current MN is within an acceptable window of recent values (window size TBD).
   b. Detect if this is an old packet by comparing the Member Identifier in the LKS Header with the stored peer state, as defined in Section 4.4.1. Note whether the peer is a LIVE PEER or POTENTIAL PEER but do not yet update the stored state.
2. Verify the ICV with the following steps:

a. Extract the CKI from the LKS Header, and verify that it is a known CKI, associated with a CA.
b. Using the CAK associated with the CKI, compute the expected ICV as defined in Section 4.3.2.2. Compare the result to the actual ICV in the message. If the expected ICV and actual ICV values do not match, abort processing.

3. Validate that the format of the message is correct. I.e., that it is a well formed message conforming to the definition in Section 4.12.
4. Update the sender's status as a LIVE PEER or POTENTIAL PEER, and its most recent MN value.
5. Check whether two distinct live peers are claiming the same SCI value. If so, a log message must be displayed.
6. Follow this logic:

```
IF sender is not a LIVE PEER
    Transition to Waiting
IF the NULL_SAK flag is set
   IF the receiver is P
      IF the CREATE_NEW flag is set
         Create new SAK, its identifier, and
            lifetime.
         Transition to Send SAK_MSG
      ELSE
         IF the sender has a new MI value
            Create new SAK, its identifier,
               and lifetime.
            Transition to Send SAK_MSG
ELSE
  IF sender is P
    IF the SAK is a new SAK
      IF the receiver is found in the
           sender's live peer list
        Install new SAK
  ELSE
    Perform election
    IF sender is new P
      IF the receiver is found in the
           sender's live peer list
        Install new SAK
  Transition to Waiting
```

NOTES

- If the sender is not a LIVE PEER, no further processing is performed on the message, even if it contains a new SAK. This stops SAKs in replayed and delayed packets from being erroneously installed.
- If a station in Primary role receives a SAK_MSG with the NULL_SAK flag, it checks if the MI value is newly added to the LIVE PEER list. If so, then it creates and distributes a new SAK. This semantic protects the receiver from breaking the AES-GCM security condition, as described below.
- A receiver of a new SAK does not install a new SAK if the sender does not yet show the receiver in its live peer list sent in the message. This check protects against a re-initialized live station from re-installing a SAK that was previously in use before it re-initialized. If the receiver had used it previously, then it cannot use the key again because it cannot know its previously used Packet Number (PN) values. The PN comprises part of the AES-GCM Initialization Vector (IV). If B were to re-use MACsec PN values, it would violate the AES-GCM security condition that a packet never be encrypted twice with the same key and IV. The receiver must wait for P to recognize that the receiver is a live peer, whereby it will generate a new SAK.
- When the station performs the election process it follows the process described in Section 4.13.3.

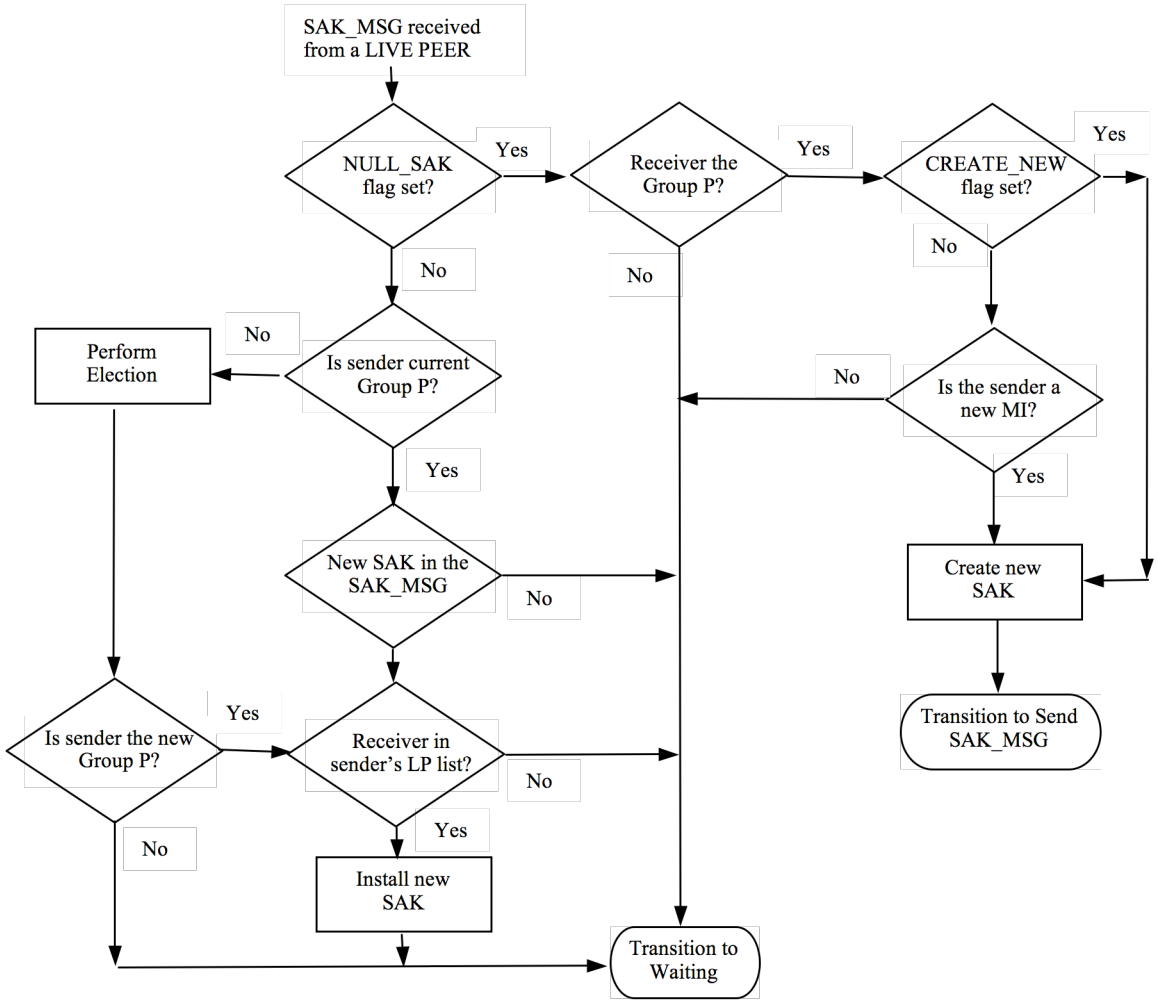



**Figure 5. Receive SAK_MSG Processing**

### 4.13.3. Election Process

When a station receives a message containing a SAK from a station that it does not currently believe to be the current group Primary, it performs an election. An election is also called when the current group Primary becomes non-responsive. The following process is followed by the station:

1. Find all stations last known to be in the P role and Compare their MI values.
   a. If more than one MI is found, choose the highest MI value to be the new Group P.
      i. If more than one station carries the winning MI value, compare the SCI values of the stations and choose the highest MI value to be the new Group P.
      ii. No two stations should ever have the same MI and SCI values.
   b. If only one MI value is found in the P role, choose it to be the new Group P.
2. If no stations are found to be in the P role, compare the MI values of all stations.
   a. If more than one MI is found, choose the highest MI value to be the new Group P.
      i. If more than one station carries the winning MI value, compare the SCI values of the stations and choose the highest MI value to be the new Group P.

## 4.14. Security Analysis

TBD

NOTES:

1. One station determines the key for the group. It would be possible for that key to act inappropriately and intentionally choose keys in some non-random manner (e.g.,

cycle through three keys known to a device not holding the CAK. However, it should be noted that there are other more insidious side channels by which a station in any role can covertly leak a key. E.g., using the key as an AES-GCM nonce in a data packet.

2. Allowing stations to choose their own MI as frequently as they like opens up the protocol to a Sybil attack[3], where a station intentionally or inadvertently chooses many simultaneous identities. This attack cannot be stopped without providing a source origin authentication, which was not a goal of IEEE P802.1af.

## 5. References


[1] E. Barker, et. al.,, "Recommendation for Key Management – Part 1:General", NIST Special Publication 800-57 Part 1, August 2005.

[2] E. Barker, et. al., "Recommendation for Key Management – Part 2:Best Practices for Key Management Organization", NIST Special Publication 800-57 Part 2, August 2005.

[3] J.R. Douceur, 2002. The Sybil Attack. In Revised Papers From the First international Workshop on Peer-To-Peer Systems (March 07 - 08, 2002). P. Druschel, M. F. Kaashoek, and A. I. Rowstron, Eds. Lecture Notes In Computer Science, vol. 2429. Springer-Verlag, London, 251-260. Available at http://www.cs.rice.edu/Conferences/IPTPS02/101.pdf

[4] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", NIST Special Publication 800-38B, May 2005.

[5] IEEE, "IEEE P802.1AE/D5.1 Draft Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security", January 19, 2006.

[6] McGrew, D. A. and J. Viega, "The Galois/Counter Mode of Operation (GCM)", May 31, 2005. Available at http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-revised-spec.pdf .

[7] NIST, "Advanced Encryption Standards", FIPS 197, November 2001. Available at http://csrc.nist.gov/publications/fips/index.html.

[8] NIST, "Annex C: Approved Random Number Generators for FIPS PUB 140-2, Security Requirements for Cryptographic Modules", Draft, January 31, 2005.

[9] NIST, "Security requirements for Cryptographic Modules", FIPS 140-2, May 2001. Available at http://csrc.nist.gov/publications/fips/index.html.

[10] NIST Computer Security Division's CSRC Home page, See http://csrc.nist.gov/.

[11] Perrig, A. et. al., "Efficient Authentication and Signing of Multicast Streams over Lossy Channels", IEEE Symposium on Security and Privacy (May, 2000), pp. 56-73.

[12] Perrig, A., et. al., SPINS: Security Protocols for Sensor Networks, Wireless Networks (September, 2002), vol. 8, num. 5, pp. 521-534. Available at http://sparrow.ece.cmu.edu/~adrian/projects/mc2001/spins-wine-journal.pdf

[13] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.

[14] Seaman M., "A distributed fault-tolerant group key selection protocol for MACsec", Revision 0.4, December 2004.


## 6. Acknowledgments


This paper was written after several visits with Mick Seaman. His help was vital for the author to understanding the principles and operations of IEEE 802.1 networks, as well as better understanding the issues surrounding implementing security in this area. Mick also suggested several optimizations that substantially simplified the LKS state machine.

For ease of comparison this paper carries forward some mechanisms and terminology from the Key Selection Protocol [14], which is authored by Mick Seaman.

David McGrew provided guidance regarding the cryptographic protections of LKS.

Joseph Salowey and Frank Chao provided many valuable insights and suggestions.


## Appendix A. Examples

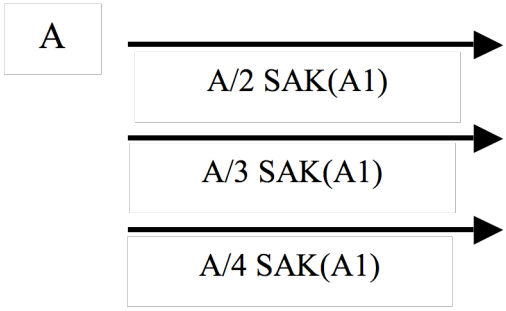
The following sections illustrate the protocol flow for LKS, including the election process.

### A.1. Two-party Exchange

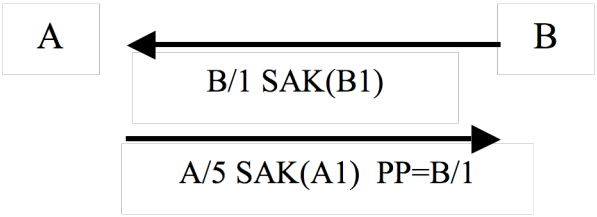
Assume a shared media LAN with only two authorized stations, and the stations do not have a synchronized boot-up procedure. The first station to boot will assume it is the Primary. It will create a

new SAK ("SAK(A1)", and provide it to MACsec. It will set its MI to "A" and initialize its MN at 1. It then begins by sending a message containing a new SAK. .


Because A receives no replies from other stations, it continues to act as P (for itself) and continues to send messages, acting as periodic heartbeat messages, as shown below.




Eventually station with MI of "B" comes on-line. It also begins in the Primary role. Since B does not yet have any live peers in its "Live Peer" list, it creates "SAK(B1)" and broadcasts it in a message., and requests a SAK. When station A receives the message the message it validates the packet and updates its "Potential Peer" list include B/1. However, because A does not yet have confirmation that B is live it does not process B's message any further. A does reply with its own message, including B/1 in its Potential Peer list..



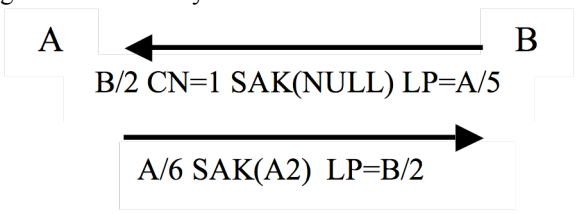

B receives and validates the authenticity of the A/5 packet. It then believes that A is alive because A reflected back its most current MI/MN pair. Station B now has to enter the election process to decide whether or not to relinquish its role as Primary. Assuming the election process declares that A should remain Primary, it accepts this decision.

However, B does not yet unwrap the SAK and give it to MACsec. B does not know whether it ever used SAK(A1) before it rebooted. If it had, then it cannot use the key again because it cannot know the list of previously used Packet Number (PN) values. The PN comprises part of the AES-GCM Initialization Vector (IV). If B were to re-use MACsec PN values, it would violate the AES-GCM security condition that a packet never be encrypted twice with the same key and IV. Therefore, B must wait until A generates a new key, which it is obligated to do after it discovers a live peer with a new MI value.

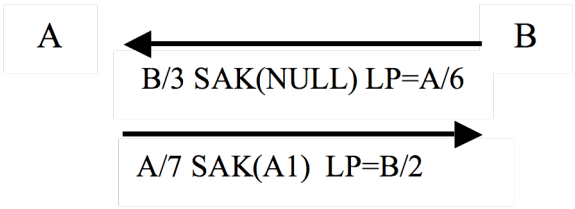
In order for A to accept B as a live peer (and therefore create a new SAK) B sends a message. Proving its liveness to A. [11]

When A receives B's message, it first observes that B includes A's most recent MI/MN values and moves B from its Potential Peer list to its Live Peer list. Since B is now evidently not in the P role (due to the NULL SAK in the packet), A recognizes that it remains the Primary device.

For safety reasons, A also recognizes that a new MI value has joined the group. This indicates that it must generate a new key.




As time progresses, both B and A issue heartbeat messages. A continues to broadcast SAK(A2) and B now broadcasts a NULL SAK to indicate that it is not in the P role.




At such time as SAK(A2) needs to be replaced, Station A will create and send SAK(A3).

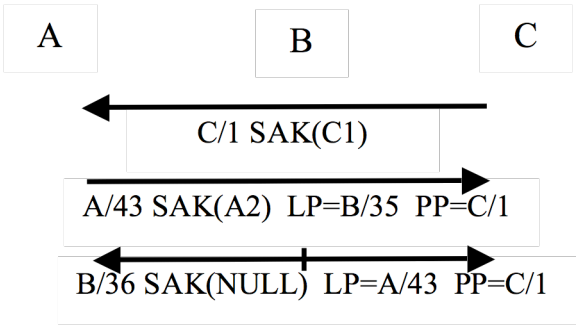

Both stations install SAK(A3), and begin using it according to MACsec semantics.

## A.2. Adding a Third party

Building on the previous example, assume that a third authorized party with an MI of "C" comes on-line. The protocol flow begins in the same manner as

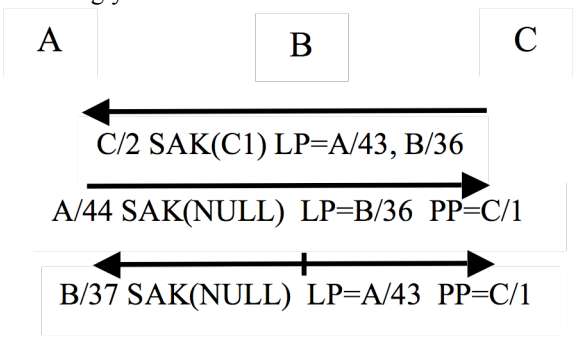
[11] For safety reasons B could request that A create a new SAK in its return packet. However this substantially complicates the state machine.

when B joined the group. B and C respond with heartbeat messages, indicating that they have received the message from C by including C/1 in their "Potential Peer" lists.




Because A's message contains SAK(A3) C discovers that A is also in the Primary role, and enters the election process. Assuming the election process declares that C should remain Primary, it does not change its state (other than to mark A and B as Live Peers).

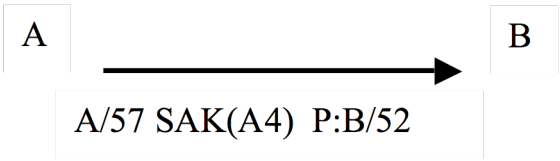
After some interval, all three stations issue heartbeat messages. Again, both A and C include their SAKs. At this time, both A and B will discover that two live stations are acting in the Primary role and will enter the election process. Because A and B now the same state as C, the election process on each will declare that C is the Primary and they will adjust their state accordingly.




C will continue to act as Primary now until there is change of stations in the group.

## A.3. Re-election of a Primary

Continuing with the previous example, assume that station C goes off-line. It will stop sending heartbeats, after which A and B will both remove it from their Live Peer lists. At this time, both A and B will observe that no station is claiming to be Primary and they will enter the election process again. Depending on its policy, the winner will either generate a new SAK immediately or simply broadcast the current SAK until it expires.




Assuming A wins, and assuming A's policy is to create a new key, A immediately creates a new SAK(A4) and sends it to the group (i.e., to B).

## Appendix B. Adding Per-sender keying support to the LKS protocol

MACsec supports the concept of each station having it's own sending key (a.k.a, per-sender key). This is shown in ([5], Figure 7-6), where three systems have three SAKs installed. In that figure each system uses a unique SAK for sending, and receives on the other two. LKS can support per-sender keys with some modest changes to its state machine.

The use of per-sender keys within a CA is advantageous for the following reasons:

- Each sender controls the duration and usage of its own keys independently.
- The key establishment function becomes straightforward because no determination must be made as to which station generates the SAK.

There are some considerations to using per-sender keys:

- Each station on the LAN must have the capacity to store one key for each station on the LAN.
- Requiring stations to be pre-configured to use a group key or a per-sender key may not be feasible. Therefore, the LKS protocol must allow stations to move between a group key and per-sender keys. If the number of stations on the LAN exceeds any one station's available key capacity, the group must revert to a group SAK.

## B.1. State Machine

It is unreasonable for devices to be manually configured in "group" or "per-sender" keys – the protocol needs to have agreement between the active entities as to which mode is to be in effect. This implies that additional protocol state is required are required. However, no additional states are required in the state machine.

### B.1.1. Protocol State

Adding support for multiple controlled ports requires additional state on each station:

- Knowledge of the number of the station's own available controlled ports
- Whether or not the group is currently using a group SAK or per-sender SAKs.

### B.1.2. Initialization State

An additional state variable must be set to indicate that per-sender keys should be used whenever possible. Also, the key capacity for each peer must be stored (with the peer's MI/MN values, etc.).

### B.1.3. Send SAK_MSG State

When a station transmits a SAK_MSG it must include its current key capacity for the CA in the message (TBD). This allows other stations to determine whether or not they are willing to use per-sender keys.

### 6.1.1. Process SAK_MSG State

When a station receives a SAK_MSG, the logic it follows in processing the SAK portion of the message is greatly simplified:

```
IF sender is a LIVE PEER
    IF the SAK is a new SAK
        Install new SAK
Transition to Waiting
```

NOTES:

1. Because the receiver never creates packets with a SAK forwarded by another station it can unconditionally install the new SAK.
2. There is no reason for a station to send a message with the NULL_PAK flag, and the CREATE_NEW flag has no meaning. Therefore, each SAK_MSG should contain the station's most current SAK.

## B.2. Frame Format

The frame format gains a couple of fields to support Per-Sender SAKs.

### B.2.1. LKS Header

The LKS Header adds the following field:

- **Number of Controlled Ports Available (2 bytes)**. This is a number that describes how many controlled ports the station is able (or willing) to devote to this CA.

### B.2.2. Specific Message Data

The SAK_MSG specific message data adds the following field:

- **SAK Mode (1 byte)**. This describes the scope of the SAK.

| *Type* | *Value* |
|---|---|
| GROUP | 0x1 |
| PER_SENDER | 0x2 |