ipdvb Working Group
Michael Noisternig
Bernhard Collini-Nocker
Internet Draft
University of Salzburg
July 14, 2008

Expires: January 2009

A lightweight security extension for the Unidirectional Lightweight Encapsulation (ULE) protocol
draft-noisternig-ipdvb-ulesec-01

Status of this Document

Copyright Notice

Abstract

The Unidirectional Lightweight Encapsulation (ULE) protocol is an efficient and extensible transport mechanism for IP over MPEG-2 networks. Such networks are often operated on broadcast wireless

channels, and are thus specifically vulnerable to attacks. Passive attacks, such as eaves-dropping, are simple to perform and emphasize the importance of security support within ULE.

This document defines a mandatory security extension for the ULE protocol that is designed with the aim of being conservative in bandwidth consumption and lightweight in the sense that it allows for implementation in low-cost, resource-scarce (mobile) receiver devices. The extension may be easily adapted to the Generic Stream Encapsulation (GSE) protocol, which uses the same extension header mechanism. The document describes the format of the security extension header, specifies default security algorithms to be used with this extension, and gives detailed processing descriptions for devices implementing the security extension.

Conventions used in this document

The following DVB specific terms are taken from [RFC4326] and recapitulated here for easy lookup:

DVB: Digital Video Broadcast. A framework and set of associated standards published by the European Telecommunications Standards Institute (ETSI) for the transmission of video, audio, and data using the ISO MPEG-2 standard [MPEG2].

MPEG-2: A set of standards specified by the Motion Picture Experts Group (MPEG) and standardized by the International Standards Organization (ISO/IEC 13818-1) [MPEG2] and ITU-T [H222].

NPA: Network Point of Attachment. In this document, refers to a 48-bit destination address (resembling an IEEE MAC address) within the MPEG-2 transmission network that is used to identify individual receivers or groups of receivers.

PDU: Protocol Data Unit. Examples of a PDU include Ethernet frames, IPv4 or IPv6 datagrams, and other network packets.

PID: Packet Identifier [MPEG2]. A 13-bit field carried in the header of TS cells. This is used to identify the TS Logical Channel to which a TS cell belongs [MPEG2].

SNDU: SubNetwork Data Unit. An encapsulated PDU sent as an MPEG-2 payload unit.

TS: Transport Stream [MPEG2]. A method of transmission at the MPEG-2 level using TS cells; it represents layer 2 of the ISO/OSI reference model.

TS Logical Channel: Transport Stream Logical Channel. In this document, this term identifies a channel at the MPEG-2 level [MPEG2]. All packets sent over a TS Logical Channel carry the same PID value.

ULE: Unidirectional Lightweight Encapsulation [RFC4326]. A protocol that encapsulates PDUs into SNDUs that are sent in a series of TS cells using a single TS Logical Channel.

Terms and abbreviations from cryptography are explained when they first appear within this document.

All numbers encoded in protocols are to be interpreted in network byte order.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL", when appearing within this document, are to be interpreted as described in [RFC2119].

Table of Contents

1. Introduction

The Unidirectional Lightweight Encapsulation (ULE) protocol [RFC4326] has been designed as an efficient and extensible encapsulation mechanism of IPv4/IPv6 and other network layer packets over the ISO MPEG-2 Transport Stream (TS) [MPEG2]. It has a simple base format, but as such does not offer any security services; however, MPEG-2 networks are often operated on wireless channels, such as satellite DVB-S [DVB-S] and terrestrial wireless DVB-T [DVB-T] and DVB-H [DVB-H] links, and are thus specifically vulnerable to attacks [ULEsec-Req]. Passive attacks, such as eavesdropping packet data or monitoring the identities (addresses) of the communicating parties, are easy to perform, and remain undetected. Low cost receiver devices and the large coverage area of satellite senders add to the likelihood of such events. Effective means to secure the ULE link are therefore important.

One solution is to rely on end-to-end security, and on one hand, reliable security can only be end-to-end. On the other hand, end-to-end security may not be applicable: this is because both sides of a communication must provide support for the same security mechanism,

which will not be realizable under many conditions where the two sides are not under central control (e.g., when browsing a public web site). One important security requirement cannot be attained by end-to-end security at all: the protection of the end-point addresses ("identities") of the communicating parties against eavesdropping (subsequently referred to as "identity protection").

By securing the ULE link only, solutions can be provided for these problems. In addition, this has the benefit that the ULE broadcast link becomes transparent for the user in the sense that he or she can rely on security assumptions as of wired links [RFC3819]. The IPsec [RFC4301] security protocols could be used in tunnel mode to create such a secure link, but this will result in significant bandwidth overhead on satellite links (due to the IP-in-IP encapsulation). Current IPsec specifications only define pairwise tunnels between two devices, thus this option is not applicable for multicast and broadcast transmissions. Last but not least, the rather high complexity of IPsec implementations might make its realization within low-cost receiver devices difficult.

Implementing security at the ULE link layer addresses above problems. A more detailed rationale for ULE link layer security and a comparison of security at the various layers can be found in [ULEsec-Req]. It also lists the security requirements for the ULE link.

This document defines a mandatory security extension for the ULE protocol that is designed with the aim of being conservative in bandwidth consumption and lightweight in the sense that it allows for implementation in low-cost, resource-scarce (mobile) receiver devices. The extension may be easily adapted to the second-generation Generic Stream Encapsulation (GSE) protocol [GSE], which shares the extension header mechanism with ULE. The format of the security extension header is described in section 2, and default security algorithms to be used with this extension are specified in section 3. These algorithms should address the most important security requirements for the ULE link: data confidentiality, identity protection, integrity protection, data authentication, and replay protection. Section 4 then gives detailed processing descriptions for devices implementing the security extension. While not defining any protocol for automated key management, some guidelines are given in section 5. After security and IANA considerations in sections 6 and 7, conclusions are presented in section 8. At the end of this document, two appendices support the reader with more insight and rationale on the decisions taken within this specification.

2. Format of the ULE Security Extension Header

This section defines the format of the ULE security extension header, ULEsec header in short. This format can be regarded as a framework for a set of security transforms. While section 3 defines default algorithms to be used within that framework, other security transforms, especially making use of other cryptographic primitives, modes, and key lengths, may be devised later and defined within separate documents.

Figure 1 below shows an example format of a ULE SubNetwork Data Unit (SNDU) containing a ULEsec header. In this example, the ULEsec extension header directly follows the base header, and it is RECOMMENDED that encapsulation devices always be configured that way. Users not following this recommendation must clearly understand the implications: first, extension headers preceding the ULEsec header cannot be protected under the data confidentiality service; second, when processing the security extension header, a receiver device may decide to discard the SNDU, a point at which preceding headers will already have been evaluated.

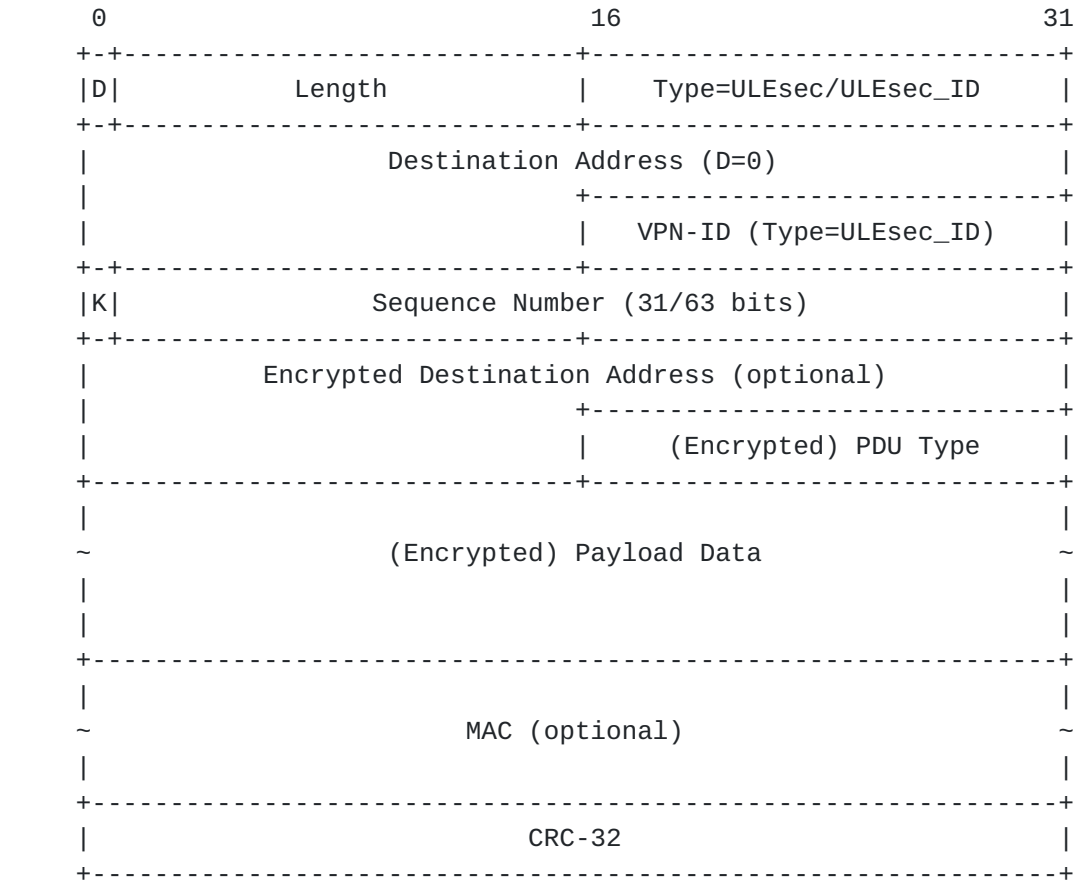


Figure 1 Example ULE SNDU containing a security extension header

The following subsections describe the fields that are part of or directly relevant to the ULEsec header. All encoded numbers are in network byte order.

2.1. Type field

The 16-bit Type field of the ULE base header (or some other extension header) indicates a security extension header following subsequently. Two different type values are defined. The first one, denoted simply ULEsec, SHOULD be used when receiver devices can uniquely identify Security Associations (SAs) based on MPEG-2 TS Program Identifiers (PIDs) and SNDU destination addresses solely. The second type, denoted ULEsec_ID, MUST be used, when PIDs and destination addresses alone are not sufficient to look up SAs. In this case, the VPN-ID field will be present, which is described next.

2.2. VPN-ID field

This 16-bit field is present when the ULEsec_ID Type is chosen. It can be viewed as a Security Parameter Index (SPI) as of IPsec implementations [RFC4301], but more adequately simply represents a Virtual Private Network (VPN) identifier. See above to decide when to use this field.

2.3. Key (K) bit

This mandatory bit provides for an easy way of detecting a key update. Whenever ULE sender (i.e., ULE encapsulator) devices switch to new keys, they flip this bit. This enables receivers to find out which of two concurrently defined set of keys (the current/old ones, or the new ones) are to be used for decoding.

New keys will be issued within key management messages by a Group Controller and Key Server (GCKS), which may or may not physically reside with a ULE sender. After each key update, devices MUST wait for a policy-defined amount of time before they permit switching to new keys again. This is necessary to avoid collisions between different keys on SNDUs sent with the same K bit. This can happen either because a receiver still accepts old keys (see section 4.3), or because a device has missed all key management messages during two periods of key updates. To avoid the latter, a GCKS may periodically send out key management messages with the key currently in use (see section 5.1).

2.4. Sequence Number

The mandatory Sequence Number field serves two purposes. First, it is part of the nonce required for the default encryption algorithm. Second, it is used for the replay protection service.

The default size of the Sequence Number field is 31 bits. This MAY be extended to 63 bits when configured as such by a Security Policy (SP) or via negotiation within a key management protocol. The larger size MUST be used when no automated key management is available.

2.5. Encrypted Destination Address field

This field is only present if the identity protection service is used (determined by the SPs selected). In that case, SNDUs do not contain a 48-bit NPA destination address in the ULE base header (i.e., they have the D bit set to 1), but the address will appear in the security extension header's Encrypted Destination Address field instead, where it will be encrypted subsequently (along with the payload data).

2.6. PDU Type field

This mandatory 16-bit field designates the type of the PDU or the next extension header in the header chain.

2.7. MAC field

The security extension header has an optional (SP-configured) trailer that follows the PDU data and contains the Message Authentication Code (MAC) of the SNDU. This MAC SHOULD have a default length of 12 octets.

3. Security Algorithms

This section specifies a set of mandatory default security algorithms to be used in conjunction with the ULEsec header. These algorithms are lightweight in the sense that the only cryptographic primitive required is the Advanced Encryption Standard (AES) [AES] with a key size of 128 bits, denoted AES-128 in short, and only its encryption part is used.

Implementation of default security algorithms is REQUIRED.

Within the following subsections, AES_mk(value) means AES-128 encryption of the 128-bit value using the master key mk, value[x..y] means taking value's bits x to y, || denotes concatenation, and x^y

means that bit x is to be repeated y times. All encoded numbers are in network byte order.

3.1. Key Derivation

In order to minimize transmission overhead within a key management protocol and to ease the setup of manual keys, separate encryption and authentication keys are derived from a single master key. The derived keys are computed as follows:

encr_key = AES_mk (Salt || 0^64)

auth_key = AES_mk (Salt || 0^63 || 1)

The Salt is a 64-bit value that MUST be an unpredictable value for adversaries. It will be transmitted along with the master key either explicitly or implicitly (e.g., derived from nonces used within the key management protocol). Including the Salt in the key derivation process preserves full security of the master key in case of compromise of any derived key against an adversary using pre-computation techniques.

3.2. Encryption

Using encryption spoils an adversary's attempt of finding out information transmitted via eavesdropping. By encoding all data following a security extension header's Sequence Number field up to but not including the MAC field, confidentiality is provided for SNDUs' payload data as well as any extension headers succeeding a security header.

Encryption is performed by employing AES-128 in the Counter (CTR) mode of operation, which is specified in [Modes], and using the encr_key defined in subsection 3.1.

The CTR mode requires a Nonce as part of its input. It is a 128-bit value and derived per packet from a 64-bit random value (Salt) that is distributed along with the master key, the 13-bit Program Identifier (PID) the underlying MPEG-2 TS cell originated from, and the ULEsec header's K bit and Sequence Number as follows:

Nonce = Salt || K || Sequence Number || 0^3 || PID || 0^16.

When 63-bit sequence numbers are used, the Nonce is computed as such:

Nonce = Salt[63..32] || Salt[31] XOR K || Salt[30..0] XOR Sequence Number[62..32] || Sequence Number[31..0] || 0^3 || PID || 0^16.

The Salt is the same as that of subsection 3.1, which primarily means that it be an unpredictable value for adversaries. Again, its purpose is to thwart pre-computation attacks.

Special care has to be taken when PID re-mapping can occur (typically within a multiplexer on a DVB network boundary [MPEG2]), as a receiver will not be able to decrypt the data successfully when using a PID value different from the sender. For one-sender scenarios where the sender also acts as the key server, a simple solution to inform receivers about such PID re-mapping may be to include the originating PID within the key management messages.

3.3. Identity Protection

For additional protection against traffic flow analysis, the ULE link layer addresses may be hidden using the identity protection service. For this, a sender omits the 48-bit NPA destination address from the ULE base header, sets the D bit, and places the address into the extension header's Encrypted Destination Address field instead, where it will be encrypted subsequently (along with the payload data). A receiver will detect an SNDU destined to it simply by probing (i.e., trial-decryption).

Identity protection has the following properties:

- There is no need to store or transmit any additional information (besides that the identity protection service is requested). Particularly, there is no need for a central server to manage or distribute addresses used specifically for this service.

- An adversary not in the know of a matching encryption key will not be able to read an SNDU's NPA destination address.

- A legitimate receiver will correctly decode the address with very high probability. In detail, the probability that an SNDU is mistakenly accepted is given approximately by k*10^-14.4, where k is the receiver's number of keys that do not match. Note that this is close to typical packet-error ratios on the ULE link for small k, which is between 10^-15.5 and 10^-16.8 on a quasi-error-free channel.

- For even lower false-acceptance rates, the authentication mechanism may be used. A MAC of size t bits will decrease the probability of erroneously accepting a SNDU with a wrong key by the factor 2^-t.

Two typical use cases for this service are sketched. In the first one, each receiver device has one distinct key to protect its unicast data. In this case, a receiver will not miss any data destined to it, and will mistakenly accept other SNDUs with negligible probability (k=1).

In the second case, all sender and receiver devices on a PID use a single shared key to protect their data, forming a VPN. Within such VPN, all devices can correctly decode all addresses (k=0).

Note that while identity protection could be used for unicast as well as multicast settings, it is sensible only for unicast communication, and as such - and in order to keep the number of mismatching keys low - should not be used for multicast scenarios.

Identity Protection MUST NOT be used without the data confidentiality service (section 3.2).

3.4. Authentication and Integrity Protection

As a mechanism against active attacks, SNDUs may carry a Message Authentication Code (MAC). A MAC provides integrity protection and source authentication for unicast connections as well as other single-sender settings. When there is more than one sender, such as in peer-to-peer settings, or when there is a possibility that a receiver in the know of the shared key might act as a sender, this mechanism gets reduced to group authentication. This is regarded sufficient, however, as attacks are primarily expected from outside (i.e., from adversaries not in the know of the right keys) [ULEsec-Req].

This construction of the MAC is based on the Cipher Block Chaining (CBC) mode of operation [Modes], and is commonly known as a (plain) CBC-MAC, which is computed as follows:

1. The SNDU, excluding the CRC and the MAC field, is first internally right-padded with zeros to an integral multiple of the cipher's block length (128 bits for AES), if necessary.

2. This padded data is then internally encrypted with AES-128 in CBC mode using the auth_key defined in subsection 3.1, and an Initialization Vector (IV) of 0.

3. The final output block of the encryption step resembles the full-length MAC whose least-significant bits are then truncated to receive the MAC of desired length.

The CBC-MAC based on AES is fully secure up to 98 bits, or about 12 octets, when used with the default sequence number space of 2^31. 12 octets is the "standard" authentication length for the IPsec protocols, and should be used as a default for ULEsec, too.

When extended (63-bit) sequence numbers are used, a block cipher with larger block size should be chosen. It is advised to take the Rijndael algorithm [Rijndael] with a block size of 256 bits as a superset of AES.

3.5. Replay Protection

Upon switching to a new set of keys, senders and receivers will set its sequence numbers to be sent or accepted next for a Security Association (SA) to the value 0. A sender will increment a sender-side sequence number by 1 after each SNDU transmitted, independently of whether replay protection is used or not. A receiver, using replay protection, will only accept SNDUs with a receiver-side sequence number higher than the last one accepted. Detailed processing descriptions regarding this service are given in section 4.

Note that replay protection using sequence numbers only works for the one-sender scenario due to the difficulty of synchronizing replay state among multiple senders. As such, this service MUST NOT be used when there is multiple legitimate senders or legitimate receivers acting as senders for a SA. Also, it SHOULD NOT be used when keys are set up manually, as a sender would have to remember its sequence number state across reboots.

4. Security Extension Header Processing

4.1. Preliminaries

Within the next subsections, the following terms are used to simplify wording:

- Basic (Policy) Selector: a pair of destination NPA address and PID value.
- Receiver-Side (Policy) Selector: a Basic (Policy) Selector with the optional VPN-ID value.
- Sender-Side (Policy) Selector: a Basic (Policy) Selector, optionally extended by higher-layer selector data, such as IP addresses, TCP ports, etc.

The term (Policy) Selector is used interchangeably for those above.

4.1.1. Security Policy Database (SPD)

Senders and receivers define policies describing the security services required or permitted for outgoing and incoming data. The collection of such Security Policies (SPs) is referred to as the Security Policy Database (SPD).

For both outgoing and incoming data, a SPD contains an ordered list of SPs. Each SP MUST contain the following information:

- A set of Sender-Side or Receiver-Side Policy Selectors (for outgoing or incoming data respectively), defining the applicability of this SP. To simplify parsing, this set MUST be encoded as a single Selector together with a Selector Mask.

- Information about the SA(s) to be instantiated by this SP. This contains:

  - A set of subsets of above Policy Selectors, downgraded to Basic Policy Selectors (i.e., only the address and PID are taken). Each subset together with the optional VPN-ID value constitutes a SA Selector, which is used for looking up or creating a Security Association (SA) within the Security Association Database (SAD) (see next section 4.1.2). To simplify parsing, a single Basic Selector Mask MUST be stored, denoted SA Selector Mask, from which the set of subsets is derived.

  - An optional VPN-ID value, part of the SA Selector. If defined, a sender MUST use this value within the VPN-ID field of the ULEsec_ID extension header type.

  - Optional Group Controller and Key Server (GCKS) data, specified by an optional destination address and a (possibly empty set of) PID(s). A device MAY use the PID(s) as a first check for legitimacy of key management messages from a certain source. When a destination address is defined, it MUST be used to contact the GCKS for membership request on receiving a protected SNDU for which this SP matches, and when the SP does not contain default key data in its first set of Security Parameters.

- An ordered list of Security Parameter sets used for instantiating a SA, sorted according to preference. A Security Parameter set MUST allow having no security services selected at all, which MUST be interpreted as sending or receiving data without protection (i.e., SNDUs without a security extension header). A sender MUST default to the first entry in the list, unless a key management protocol permits negotiation (e.g., for unicast, bidirectional settings) and a receiver contacts the GCKS to request another set of Security Parameters from the list. Each set of Security Parameters MUST contain information about:
  - The cryptographic algorithms used.
  - The cryptographic parameters required by these algorithms (e.g., the MAC length).
  - The length of the sequence number field.
  - Optional key data for manual keying: a master key, and an optional Salt.

SPs may be manually set up by the owner of the sender or receiver equipment, or dynamically distributed via a GCKS (using a key management protocol). While the resulting SPD may become complex by containing separate SPs for each pair of PID and NPA address data may be sent to or received from, in general it is expected to contain just a few entries.

This document does not define how to store, manage, and look up SPs within the SPD, as this is regarded implementation specific details.

4.1.2. Security Association Database (SAD)

A Security Association (SA) is an instantiation of a SP. It describes the current state of a secure connection between two or more devices. All devices sharing a SA are part of the same VPN. The set of SAs of a device is aggregated in the Security Association Database (SAD).

A SA MUST contain the following information:

- The SA Selector derived from the instantiating SP.
- Any GCKS data defined by the SP and the GCKS.
- Static security parameters defined by the SP (cryptographic algorithms, MAC length, Sequence Number length, etc.).

- Current and prospective dynamic security parameters (keys, Salt, etc.), defined by the SP or the GCKS.

- The current sender-side K bit and sequence number for transmitting data.

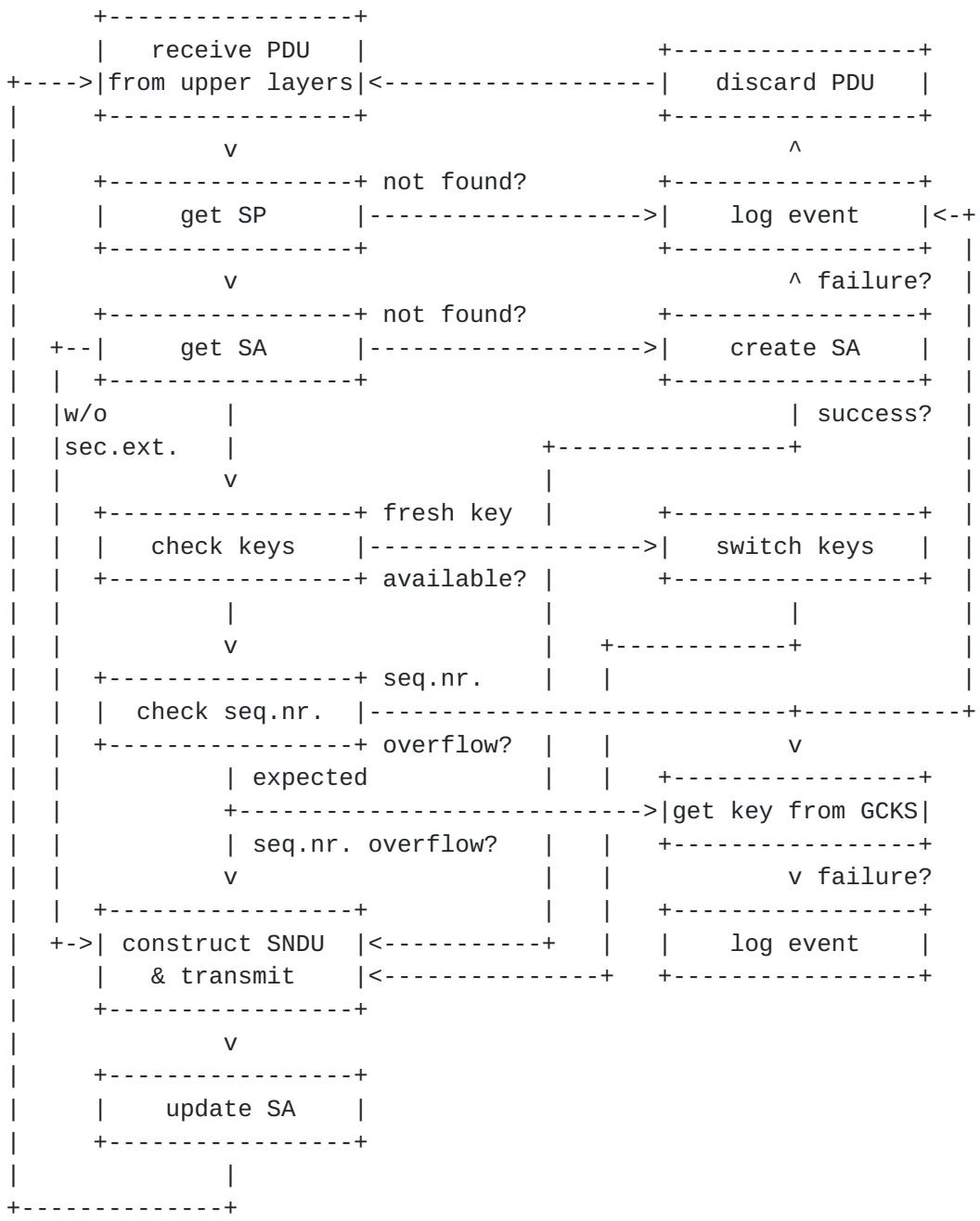
- The current receiver-side K bit for receiving data, and the current receiver-side sequence number for receiving data with replay protection.

- A flag defining whether prospective security parameters have been received through a GCKS.

As with the SPD, this document does not define how to store, manage, and look up SAs within the SAD.

4.2. Sender Processing

4.2.1. General Activity Diagram



4.2.2. Detailed Processing Description

The following list describes the processing steps for a ULE encapsulator implementing the ULE security extension (ULEsec sender in short):

1. Get SP: After receiving a PDU from upper layers for transmission over the ULE link, a ULEsec sender MUST consult its SPD by scanning the ordered list of outgoing SPs until it finds a matching policy. That is, it looks for a SP for which

   (SP's Sender-Side Selector AND SP's Selector Mask)
   == (SNDU's Sender-Side Selector AND SP's Selector Mask)

   is true. If no such policy can be found, the data MUST be discarded, and this event SHOULD be logged as an invalid transmission attempt.

2. Get SA: With a SP chosen, a SA Selector is constructed as a triple:

   SA Selector := (SNDU's Basic Selector AND SP's SA Selector Mask, SP's SA Selector Mask, SP's VPN-ID value)

   The VPN-ID value, if not defined by the SP, must be set to a reserved "null" value (i.e., a fixed value not within the 16-bit number range of the extension header's VPN-ID field).

   The SA Selector is then used to look up a SA within the SAD. If no SA is found, it must be set up as follows: If the SP's first Security Parameter set either contains default key data (master key, optional Salt, etc.) or defines data to be sent without protection, the SA is immediately created and initialized according to these settings. Otherwise, if the SP defines a GCKS destination address, the server MUST be contacted for obtaining key material. During that attempt the sender SHOULD postpone or discard transmission of the data. Any case of failure MUST result in the data being discarded, and this SHOULD be logged accordingly (e.g., as a user authentication failure in case of membership denial by the GCKS).

3. Check keys: Whenever a SA is provided with fresh key material, a sender MUST switch to the new set of keys after a policy-defined length or point of time, and prior to a sequence number overflow (see next step). This is done by flipping the SA's sender-side K bit and resetting the sender-side sequence number to 0, while selecting the fresh key material as the new current one for sending. Note that a SA that is also used for receiving SNDUs may still require the older set of keys as a receiver-side K bit will be flipped at a later (policy-defined) point of time. This is to compensate differences in key update times of multiple senders, which means there will be a period during which some devices will

already send with new keys while others will still use the old ones.

4. Check sequence number: An implementation MUST NOT allow a SA's sender-side sequence number to overflow. For a SA defining a GCKS destination address, an implementation MUST contact the server for obtaining fresh key material in anticipation of this event. When keys are set up manually, the user SHOULD be warned about an expected overflow. Should eventual transmission of an SNDU ever result in the sequence number to overflow, the data MUST be discarded instead, and this event SHOULD be logged as a sequence number overflow event.

5. Construct SNDU: For a SA that allows passing data unprotected, the SNDU is constructed as usual. Otherwise, it is built as follows:

   a. First, the ULE base header and any extension headers preceding the security extension header are written. If the SA requests identity protection, the destination NPA address MUST be omitted from the base header (with the D bit set to 1). If a VPN-ID value is defined within the SA, the last extension header's (or base header's) Type field MUST contain the value for a ULEsec_ID extension header; otherwise, it contains the ULEsec extension header value.

   b. For the ULEsec_ID extension header, the 16-bit VPN-ID field is written.

   c. Next, the SA's sender-side K bit and sequence number are filled into the extension header's mandatory K Bit and Sequence Number fields. The length of the Sequence Number field is defined by the SA.

   d. For the identity protection service, the destination NPA is encoded as defined by the SA, which means it will be encrypted along with any subsequent extension headers and the payload data for the default identity protection algorithm.

   e. Subsequently, the mandatory (Encrypted) Type field, any other extension headers, and the PDU are encoded as defined by the encryption algorithm selected.

   f. For authentication, a MAC of length as defined by the SA is

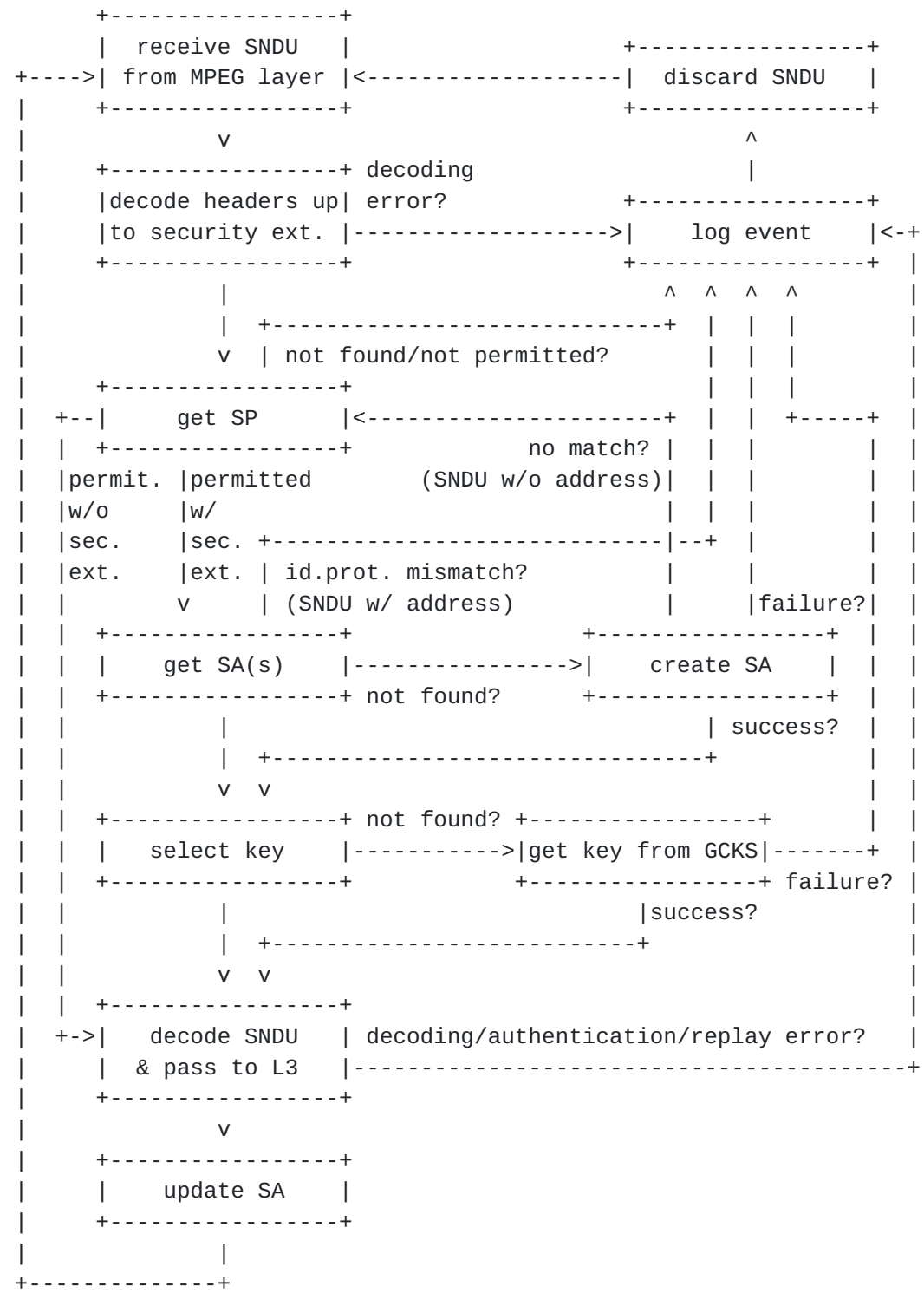
appended. The MAC is computed over all the data encoded so far, which means, from the start of the SNDU to the end of the payload data.

g. Finally, the CRC is calculated and appended, and the SNDU further processed according to [RFC4326].

6. Update SA: After processing a protected SNDU is completed, a sender MUST increment the SA's sender-side sequence number by 1.

4.3. Receiver Processing

4.3.1. General Activity Diagram



4.3.2. Detailed Processing Description

A receiver implementing the ULE security extension (ULEsec receiver in short) must follow below procedure upon reception of a ULE SNDU:

1. Decode SNDU (1): The SNDU is checked for a correct CRC as described in [RFC4326], after which the base header and the extension headers up to either a security extension header or the PDU are evaluated. From the base header, a Basic Policy Selector is constructed. This one is extended to a Receiver-Side Policy Selector by adding the VPN-ID field of a ULEsec_ID Type security extension header, when encountered.

2. Get SP: After the SNDU passed the first filtering and evaluation step, the SPD's ordered list of incoming policies is scanned for a matching policy.

   a. D=0: With the SNDU's D bit cleared, the following check must be true for selecting a SP:

      (SP's Receiver-Side Selector AND SP's Selector Mask)
      == (SNDU's Receiver-Side Selector AND SP's Selector Mask).

      If no matching policy can be found, the data MUST be discarded immediately, and this event SHOULD be logged as reception of an invalid SNDU.

   b. D=1: When the D bit is set, above check is done as well, except that the destination NPA address values are ignored.

      If no matching policy can be found, the data MUST be discarded immediately, and this event MAY be logged as reception of an invalid SNDU.

   If the SNDU is received without a security extension header but the SP does not permit unprotected data to pass, the SNDU MUST be discarded immediately, and this event SHOULD be logged as reception of an invalid SNDU. Likewise, if there is a security extension header but the policy allows only for unprotected data, the SNDU MUST be discarded, and this event SHOULD be logged.

   When permitted, an SNDU without a security extension header is decoded as usually. For a protected SNDU, processing continues with step 3.

3. Get SA: With a first match on a SP, a SA Selector is constructed to find a SA within the SAD.

a. D=0: With a destination address present, the following SA Selector is used to directly look up a SA within the SAD:

SA Selector := (SNDU's Basic Selector data AND SP's SA Selector Mask, SP's SA Selector Mask, SNDU's VPN-ID value)

The VPN-ID value must be set to a reserved value if not defined by a ULEsec_ID header.

If no SA is found, it must be tried to set it up as described in step 2 of section 4.2, Sender Processing. If creation of a SA fails, the SNDU MUST be discarded, and this SHOULD be logged accordingly.

b. D=1: With the D bit set, a set of SAs is looked up by omitting the destination address:

SA Selector := (SNDU's PID AND SP's SA Selector Mask's PID, SP's SA Selector Mask, SNDU's VPN-ID value)

From the retrieved set, every SA not defining identity protection is ignored. Assuming the remaining set is not empty, the K Bit and Encrypted Destination Address field are read ahead from the security extension header. The current key, as defined by the K Bit (see next step), is then taken from the first SA in the set to trial-decrypt the Encryption Destination Address field (i.e., it is decrypted to a temporary buffer). The decrypted address MUST be checked to match both the SP selected as well as belong to the current SA. Then, it MUST be compared with all destination NPA addresses a receiver accepts to look for a match. If either a SA does not contain the current key, or there is no match for an address, the SA is removed from the retrieved set, and probing is done with the next one.

If no SA matches, but the SP defines default key data as well as the identity protection service for the first set of Security Parameters, the encryption key derived from the default key data is used for probing with the SNDU as described in above paragraph. (Because of this, receiver-side SPs containing default key material SHOULD already provide derived keys for efficiency reasons.) If this test succeeds, a SA is constructed using the SP's first Security Parameter set; if SA creation fails (e.g., due to out-of-memory conditions), the SNDU MUST be discarded, and this SHOULD be logged accordingly.

When no matching SA can be found, the SP is assumed to be selected mistakenly, and processing MUST continue at step 2.b with the next SP in the list of incoming SPs.

4. Select key: Once a matching SA is found, proper keys must be looked up. For this, the SNDU's K bit is compared with the SA's receiver-side one. If they are equal, the SA's current keys are taken for decoding. Otherwise, the SA's prospective keys must be selected. If these are not defined, the SNDU MUST be discarded and this event SHOULD be logged as reception of an invalid SNDU. A receiver SA, when provided with prospective keys, must switch to these after a policy-defined point or amount of time by flipping its receiver-side K bit and replacing the current keys with the fresh ones.

5. Decode SNDU (2):

a. Authenticate SNDU: For verifying authenticity and integrity, a MAC is computed as defined for the sender side, and then compared with the value of the SNDU's MAC field for equality. If the two values differ, the SNDU MUST be discarded, and this SHOULD be logged as a data authentication failure.

b. Replay Protection: To detect replays, the SNDU's Sequence Number MUST be greater than or equal to the SA's receiver-side sequence number; if this is not the case, the SNDU MUST be discarded, and this event SHOULD be logged as a replay.

c. Decryption: If the data confidentiality service is used, all data starting from the security extension header's Encrypted Type field up to the end of the PDU data are decrypted. After that, processing continues as normally (i.e., any other extension headers are evaluated, and the PDU is finally passed to upper protocol layers).

6. Update SA: Now that the SNDU has been accepted, a SA using replay protection must be updated accordingly: If the K bits of step 4 were differing, keys are switched immediately as described in that step; then, the SA's receiver-side sequence number is set to the SNDU's Sequence Number value, and incremented by 1.

5. Key Management Considerations

Manual key setup is simple but practical only for small and relatively static secure groups. When number of receivers gets high, or users need to be added or excluded more frequently, automated key

management becomes necessary. A Group Controller and Key Server

(GCKS) uses a key management protocol to automatically distribute key material to legitimate devices in the event of new membership, revoking of users, or key update (either periodically for increased security, or after a security breach). The definition or selection of any particular key management protocol is out of scope for this document as doing so has no influence on extension header format, security algorithms, or extension header processing. However, some important considerations are discussed next, and one must distinguish between the two different scenarios of unidirectional and bidirectional communication.

5.1. Unidirectional Key Management

In unidirectional settings, security parameters can merely be decided by the sender. Having no way for negotiation, this allows for a simpler protocol design in this regard. However, other issues arise. Senders must assure reliable delivery of information to all receivers. Doing so might require sending the same data multiple times. Receivers must be able to jump into a session at any time and without substantial delays, either when they are turned on, or after loss of synchronization. Replay attacks must be considered. They are best countered with timestamps; however, this requires sender and receivers to have synchronized clocks.

The design of a unidirectional key management protocol should allow installing, updating, and revoking a number of different keys within receiver devices, with new keys encrypted with any other one. Different keys can correspond to individual, group, or global keys. This flexibility will allow the use of various broadcast encryption algorithms (such as the subset difference scheme [Subset]).

It is further suggested that a unidirectional key management protocol uses the same format for re-key messages as for the initial key distribution. In other words, re-keys should provide all the information necessary to allow receivers to jump into the middle of a session, which shall result in great aid for connectivity.

Existing mechanisms should be evaluated, such as [DVB-CA] and [ATSC-CA]. For example, [DVB-CA] defines unidirectional key management in the form of the entitlement checking and entitlement management messages. This offers a simple mechanism for securely establishing, updating, and revoking keys. A 3-level key hierarchy is used to provide for a better level of safety in case of key compromise. However, the single first-level key remains unchanged, constituting a weakness in this system; this could be mitigated in a protocol for the ULE security extension by either configuring receiver devices with unique sets of first level keys, thereby allowing true broadcast

encryption algorithms such as [Subset], or by permitting operators of a VPN to update first level keys externally of the ULE network (either manually or automatically).

5.2. Bidirectional Key Management

The availability of a back channel allows devices not only to actively request group membership, revocation, and retransmission of keys after loss of synchronization. Negotiation of cryptographic algorithms and keys becomes possible, as well as enhanced security features such as perfect forward secrecy, mutual authentication, and identity protection when receivers are identified by other means than their NPA address.

In contrast to unidirectional key management, numerous bidirectional protocols have been developed for various layers of the ISO/OSI reference model. Prominent examples include [DVB-RCS] (link layer), IKEv2/IPsec [RFC4306] (network layer), TLS [RFC4346] (transport layer) and SSH [RFC4253] (application layer). Naturally, they differ highly in purpose, functionality, and complexity. While existing link layer technology such as those within [DVB-RCS] is probably most directly usable for ULE, requirements for bidirectional key management must be clearly determined.

Traditional key management protocols, including the ones cited above, are designed for unicast communication, only. ULE key management must support VPN-like settings with a potentially large number of receivers. One focus of the IETF MSEC working group is on developing and standardizing scalable solutions for key management within large groups, and several different protocols have been proposed [RFC4535, RFC3547, GKDP, FMKE, RFC3830]). Clearly, these must be considered when defining ULE key management.

6. Security Considerations

The security of cryptography-based systems depends in one part on the strength of the cryptographic algorithms chosen and the strength of the keys used with those algorithms. The algorithms identified in this document are not known to be broken at the current time, and research so far leads to believe that the combination of algorithms and key lengths specified within this document will likely remain secure into the foreseeable future. Considerations that relate to this aspect, including the correct use of algorithms, are addressed in their respective sections or the appendices of this document.

There is a caveat regarding the security extension's Sequence Number field when a SA secures communication for a single receiver device

(i.e. for unicast communication) and the identity protection service is used. A passive attacker may link SNDUs with increasing sequence number to the same SA, thereby increasing its chance of identifying a receiver and the amount and type of data transmitted to it. Future research will investigate on possible solutions for this problem.

The security also depends on the engineering and administration of the protocols used by the system to ensure that there are no non-cryptographic ways to bypass security. When selecting a key management protocol, its security will be a pre-requirement for overall system security.

ULE link layer security may not be treated as a replacement for end-to-end security. If reliable security is required, one MUST use end-to-end security protocols. However, ULE link layer security can complement end-to-end security as laid out in the conclusions in section 8.

7. IANA Considerations

This document requires two 16-bit Type codes to be registered by the IANA, namely one for the ULEsec security extension, and one for the ULEsec_ID extension header type. It is suggested that the two Type codes are allocated in a way that they differ only in their least-significant bit. This allows use of this bit as a flag for determining the presence of the VPN-ID field. (However, this is merely an optimization and no requirement.)

8. Conclusions

The solution presented within this document addresses many of the security requirements for the ULE protocol laid out in the separate security requirements document. Passive attacks, constituting the most important threats in the ULE network, are effectively defeated using the data confidentiality and identity protection service. To mitigate active attacks, MACs may be used to assure a receiver of the authenticity and integrity of the data received. While not providing source authentication for VPN-like settings with multiple senders, they still render outsider attacks futile. Sequence numbers within each SNDU allow for simple detection of replayed data on unicast connections, without any additional bandwidth overhead.

The format of the security extension header generates minimal bandwidth overhead, is extensible, and acts as a framework for a set of security transforms that may be changed and updated independently of each other. Default algorithms are defined to be lightweight and allow implementation in low-cost devices.

A key management protocol may be added later and independently of this specification. This will allow more flexibility and security in setting up secure connections and VPNs. Existing protocols such as those following the guidelines of [RFC4046] might be used. However, this will need careful assessment regarding the applicability for the ULE link.

Note that the presented ULE security extension secures the ULE broadcast link, only, and as such may not be treated as a replacement for end-to-end security. In fact, ULE link layer security is an additional security mechanism that complements end-to-end security [ULEsec-Req]. Where end-to-end security is used, it can provide identity protection over the ULE link in addition. When the ULE link is used to directly connect two secure sites, ULEsec can be the sole provider of security. In the cases where end-to-end security is not applicable, the ULE security extension can be viewed as protecting the weakest link, and a user can rely on security assumptions as of wired links.

9. Acknowledgments

The document was prepared using 2-Word-v2.0.template.dot.

APPENDIX A: Rationale for the Extension Header Format

A.1. (16-bit) optional VPN-ID vs. (32-bit) SPI

In order to identify which secure connection (represented by a SA) a secured packet belongs to, it is generally marked with a connection identifier. For the IPsec [RFC4301] security protocols, this identifier is known as the Security Parameter Index (SPI). It is selected by the receiver, only, in order to avoid collisions between different connections. This works because IPsec is designed for secure unicast communication only. For multicast settings, a manually assigned SPI together with manual keying can be used. The same principle of deriving a secure connection identifier can be used with the ULE security extension, too. Instead of an SPI, the ULE security extension provides the VPN-ID field, when the ULEsec_ID security extension header type is used.

One difference between IPsec and ULEsec is that IPsec is an end-to-end security protocol. This means that on a single ULE link a ULE decapsulator will potentially accept data from many different (IPsec) end-to-end connections. In contrast, at the ULE link layer users are expected to establish just a single or otherwise so few secure L2 connections that in many cases receivers can even identify connections based on ULE destination NPA addresses and MPEG-2 TS PID values alone. Therefore, a size of 16 bits has been chosen for the ULEsec VPN-ID field as opposed to 32 bits as used for the SPI of the IPsec security protocols. In addition, it is provided for a way to omit the VPN-ID field altogether (by selecting the ULEsec extension header type).

A.2. VPN-ID + K-Bit vs. SPI

The way re-keying works in the IPsec protocols is by creating a new SA (i.e., secure connection) for the new keys. The new SA will have a different SPI value than the old one, selected by the receiver again. This model does not work for ULE, however. A receiver that failed to receive key update messages will have no way of determining that the new data with a different secure connection identifier is to be received by him. This is not only an issue for multicast settings where scalability is a concern, but also for unidirectional links where there is no way for feedback. Instead, ULEsec uses the K Bit to signal when re-keying occurred, and keeps the VPN-ID value constant. Now a receiver will always know which data to accept, and it would have to lose key management messages during two periods of key updates for this model to fail.

A.3. 31-bit/63-bit Sequence Number

A 31-bit sequence number has been chosen as a default as it does not add too much overhead and is reasonably big for automated re-keying to happen infrequently. For example, when SNDUs are continuously sent on a link with an effective bit rate as high as 68 Mbit/s [DVB-S], protected under the same key, and contain only TCP/IP acknowledgments so the SNDU size is just 60 octets (20 octets ULE+ULEsec header, 20 octets IP header, and 20 octets TCP header), then a key can still be used for more than four hours before the 31-bit sequence number space will be exhausted.

For high-speed links, and when manual keying is used, the larger 63-bit sequence numbers are to be used. This pairs well with a key size of 128 bits for the encryption algorithm, where after 2^63 uses security will be about halved and new keys should be set up by then.

While the Sequence Number field could be optional, too, its placement has been made mandatory in order to not unnecessarily complicate things, and since it is required virtually always, anyway. It could be replaced with a timestamp, though, but this is not defined within this specification.

A.4. MAC field

The MAC field is not a direct part of the security extension header, but defined as a trailer. This allows the MAC to be computed in an online way. For example, a sender can compute the MAC of an SNDU while it is already transmitting it, and when it has sent the last bit of the payload it can simply attach the MAC.

For a similar reason, the CRC, which can be viewed as part of the ULE base header, is at the end of the SNDU.

APPENDIX B: Rationale for the Default Security Algorithms

B.1. Encryption

The first question on the data confidentiality service is whether to use a block cipher or a stream cipher algorithm. Stream ciphers offer the benefit of allowing for very simple and fast implementations. One particular stream cipher is the Arcfour (or RC4) algorithm [Arcfour], and it can be regarded the de-facto standard among software based implementations. However, several weaknesses have been found so far, and while there are ways to circumvent some of them, they do not make Arcfour a favorable choice [Arcfour-Fix]. Unfortunately, other stream ciphers available are either not sufficiently analyzed, or are based on linear feedback shift registers, making them suitable for cheap hardware implementations but vulnerable to algebraic attacks.

Among block ciphers, things are looking much better. The Advanced Encryption Standard (AES) [AES] has been quickly embraced as a secure, fast, and open encryption algorithm since its election within a competition held by the American NIST in the year 2000 to replace the aging DES. Despite algebraic structures found, AES is still considered secure.

A mode of operation is required for a block cipher to allow it to be repeatedly used securely under the same key. NIST has standardized several such modes [Modes]. One particular mode is the Cipher Block Chaining (CBC) mode. It is well-understood and has good security properties. However, it operates on full blocks only, and data must therefore be padded to multiples of block lengths in general. In addition, the CBC mode requires an explicit (pseudo-)random Initialization Vector (IV) of the size of a cipher block for each packet. This is clearly wasteful for the ULE scenario, where this means additional average overhead of 24 octets per SNDU when used with AES.

The Counter (CTR) mode in contrast effectively turns the block cipher into a stream cipher, so there is no need for padding. IVs for this mode come as nonces, so a simple counter may be used. A counter can not only be encoded in less space than the size of a cipher block, it may also serve as a mechanism for replay protection - in which case no space will be wasted at all. Some of the other advantages of the CTR mode are that only the encryption part of the block cipher algorithm is needed, and both encryption and decryption can be parallelized.

Strong care must be taken for the nonces to never be used twice under the same key, or all security will be lost. Therefore, within the

construction of the nonce, not only the SNDU's sequence number but also the MPEG-2 TS cell's PID value is included. This gives a guarantee for nonces to be unique when the encryption key is shared across more than one PID. One case is that of multiple senders (i.e. ULE encapsulators) using the same keys. It is assumed that there will be at most one sender per PID for technical reasons (this does not preclude the possibility of adversaries sending on the same PID). A similar scenario is where a sender spreads data over multiple PIDs.

The last question is what key size to use. AES may be used with key sizes of 128, 192, or 256 bits. Of these, 128 bits are regarded to be sufficiently secure even for the foreseeable future, and any bigger size would merely result in increased key management overhead and computational effort [Standards].

B.2. Identity protection

The presented solution for identity protection is simple, yet very effective. First, a receiver needs to probe only a very low number of keys with an SNDU. This is because in the vast majority of cases there is only a single or otherwise very few different keys a receiver associates with a PID or pair of PID and VPN-ID field: For an incoming ULEsec packet with a VPN-ID field present, the pair of PID and VPN-ID will most commonly denote a single VPN with a single shared key. Without a VPN-ID, the PID may represent a VPN with a single key, or the PID will probably contain several or many unicast or multicast connections of which the receiver accepts only a few, namely for its own unicast and a few (if any) multicast addresses. Each of these connections could utilize identity protection; however, this is sensible only for unicast communication. Assuming a device is not assigned more than one unicast address, there is only a single key left that has to be probed with.

Second, by encrypting the address an adversary not in the know of a matching decryption key will not be able to read the packet's destination address. A legitimate receiver, in contrast, will correctly decode the address with very high probability. In detail, the chance that an SNDU is mistakenly accepted (assuming the encryption algorithm behaves as a pseudo-random function under different keys) is given approximately by k*10^-14.4, where k is the receiver's number of keys that do not match. This is close to typical packet-error ratios on the ULE link for small k, as this example shows: assuming a quasi-error free channel with a bit-error ratio of 10^-10 [DVB-S], and - for simplicity assumed - a probability of 2^-32 for the CRC-32 failing, the chance of receiving an erroneous SNDU undetected by the CRC ranges between approximately 10^-15.5 for a

typical 1500-octet (file download) payload and 10^-16.8 for small (VoIP) data of 60 octets.

Note that this method of identity protection requires to be combined with the data confidentiality service for two reasons: First, it protects only L2 addresses. Second, there is a requirement for the data a receiver assumes to be an encrypted destination address to be pseudo-random, or at least non-repeating (with high probability), otherwise above equations will not hold.

This solution for identity protection is also superior to other suggestions such as the use of temporary destination addresses [ULEsec-CPI] for a variety of reasons. First, when creating temporary addresses it must somehow be assured that these do not and will not collide with real-world addresses, i.e. addresses that are in use or might be used in the future. Second, in order to assure these addresses to be unique a server must keep every one of them in a database. This is clearly a disadvantage for simple settings such as VPNs where otherwise only a single key must be stored. Third, such addresses compromise additional information that must be distributed in a reliable way, which is difficult for unidirectional links and does not scale well for multicast scenarios. Last but not least, even though these addresses are temporary, they remain constant for a period of time and as such may pose just another chance for an adversary to link packets with a receiver.

B.3. Authentication

By adding redundancy in the form of keyed cryptographic checksums to packets, legitimate receivers can verify both authenticity and integrity of the data. A Message Authentication Code (MAC) is the result of a symmetric checksum function, so both senders and receivers use the same key for authenticating and verifying.

MAC algorithms typically build upon cryptographic hash functions (message digests), such as MD5 [RFC1321], SHA-1 and the SHA-2 family [SHA], and RIPEMD-160 [RIPEMD-160]. The HMAC [RFC2104] is a popular construction to turn a hash function into a MAC function. The advantage of using hash functions is their simple implementability, resulting in certain speed advantages. Despite a number of surprising and unexpected collision attacks on hash functions published lately [MD5-Attack, SHA-1-Attack], the HMAC construction is secure as long as no second pre-image attacks become practical, and the hash function's output cannot be distinguished from random data by an adversary [Preimages, HMAC2, Standards].

MAC constructions may also use block ciphers as building blocks. The main advantage of this approach is that an already existing block cipher implementation can be re-used. This makes the CBC-MAC (and its variants) still a very popular solution. The security of a CBC-MAC is directly dependent on that of the block cipher. Because of the birthday phenomenon, the upper limit of security is also determined by the block length of the underlying encryption algorithm, which degrades quadratically over time [CBCMAC1, CBCMAC2]. Therefore, when the same key is used for a long time (e.g., with extended sequence numbers for ULEsec), a cipher with a higher block length should be chosen, such as [Rijndael] with a block length of 256 bits.

A plain CBC-MAC is not a generally secure construction. In detail, it is only secure for fixed-size or prefix-free messages [CBCMAC1]. However, ULE SNDUs are automatically prefix-free because of the inclusion of the length field in the base header.

There are some interesting newer constructions based on Carter-Wegman universal hashing, such as the UMAC [RFC4418] and the [Poly1305-AES], both defined for use with the AES encryption algorithm. While having excellent security properties, allowing for parallelization, and achieving high throughputs on modern desktop processors, they are not suitable for small hardware devices because of performing complex operations such as multiplications in large size Galois fields and requiring a large amount of memory.

Two surprisingly simple and parallelizable designs are presented in [XOR-MAC] and [PMAC]. When a block cipher is plugged into the pseudo-random function required, the complexity is similar to that of the CBC-MAC. Security is not affected by the birthday phenomenon, however. As both algorithms are covered by patents, they are not selected as default authentication algorithms.

B.4. Source Authentication

MAC algorithms are symmetrical functions, so anyone in the know of the right key could have been the author of an authenticated message. Consequently, MACs cannot provide source authentication when there is more than one legitimate sender, or receivers are able to act as senders. In order to guarantee data coming from the source as corroborated, some asymmetry must be introduced, either by using functions that are hard to invert (digital signatures), or by disclosing verification key material only after it has been used for authentication (e.g., TESLA [RFC4082]).

Source authentication does not come for free, however. Digital signature algorithms are very computationally demanding, as time

needed for signing and verification is still counted in milliseconds even on modern hardware. Additionally, bandwidth consumption is high with typical key sizes of 1024 bits and signature sizes of 320 bits for a level of security comparable to that of a 80-bit symmetric key [Recommendations]. A number of different solutions have been developed over time to overcome these shortcomings, most prominently TESLA; however, all of them have one or another drawback such as increased latency at the sender or receiver side, lack of re-synchronization ability in case of packet loss, or even higher bandwidth cost.

Source Authentication for ULEsec may be devised independently of this specification, but it likely makes more sense for control messages (e.g., key management messages).

B.5. Combined Authentication and Encryption

While there had always been a lack of consensus in cryptography and security communities about the "right" way of combining authentication with encryption, it had not been know until recently that the only generally secure way of generic composition is Encrypt-then-Authenticate (EtA) [Order-AE]. By following that finding within this specification, encryption and authentication algorithms can be changed and updated independently of each other without compromising overall security; for example, the default CBC-MAC for authentication could safely be replaced with a MAC using a cryptographic hash function, or with an algorithm providing source authentication for multi-sender scenarios.

Another recent development in cryptography are so-called Authenticated Encryption (AE) and Authenticated Encryption with Associated Data (AEAD) schemes. These can be viewed as modes of operation that integrate both authentication and encryption functionality. Excitement for these schemes can primarily be contributed to the development of encryption modes that provide authentication essentially for free (i.e., with negligible computational overhead) [IAPM, XCBC, OCB]. Unfortunately, all these so-called 1-pass schemes have patents filed on them, which is an issue for an open standard. This circumstance initiated the development of second-generation unpatented 2-pass schemes, most notably CCM [CCM]. Even though these modes provide a secure two-in-one solution (one key for both encryption and authentication), they have lost their predecessors' main advantage: they are, as their name says, 2-pass. As a result, they do not offer any real benefits over the more versatile AtE generic composition approach.

B.6. Replay Protection

The key idea in providing replay protection is to guarantee each packet to be unique under the same key. This is generally achieved by adding a monotonically increasing counter or a timestamp to the packet header. The in-order delivery of data on the ULE link then allows for easy detection of replays on the receiver side.

A counter is simple to use, requires minimal connection state on each side, and is fully reliable for unicast connections and other one-sender scenarios. It cannot be used when a key is shared among multiple senders due to the difficulty of synchronizing replay state.

A timestamp uses synchronized clocks for the replay state. A small window of accepted timestamps is required to compensate timing discrepancies. This way, timestamps can be used with any number of senders. However, this also means that they are not completely reliable; consequently, their use is not defined within that specification.

10. References

10.1. Normative References

[RFC4326] G. Fairhurst, B. Collini-Nocker, "Unidirectional Lightweight Encapsulation (ULE) for Transmission of IP Datagrams over an MPEG-2 Transport Stream (TS)", IETF RFC 4326, December 2005.

[MPEG2] "Information technology - generic coding of moving pictures and associated audio information systems, Part I", ISO 13818-1, International Standards Organization (ISO), 2000.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, IETF RFC 2119, March 1997.

[AES] "Advanced Encryption Standard (AES)", FIPS PUB 197, National Institute of Standards and Technology (NIST), Nov. 2001.

[Modes] M. Dworkin, "Recommendation for Block Cipher Modes of Operation - Methods and Techniques", SP 800-38A, National Institute of Standards and Technology (NIST), Dec. 2001.

10.2. Informative References

[H222] "Information technology, Generic coding of moving pictures and associated audio information Systems", H.222.0, International Telecommunication Union (ITU-T), 1995.

[DVB-S] "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite systems", ETSI EN 300 421, Aug. 1997.

[DVB-T] "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television", ETSI EN 300 744, June 2006.

[DVB-H] "Digital Video Broadcasting (DVB); Transmission system for handheld terminals (DVB-H)", ETSI EN 302 304, Nov. 2004.

[ULEsec-Req]H. Cruickshank, P. Pillai, M. Noisternig, S. Iyengar, "Security requirements for the Unidirectional Lightweight Encapsulation (ULE) protocol", draft-ietf-ipdvb-sec-req-08 (work in progress), July 2008.

[GSE] "Generic Stream Encapsulation (GSE) Protocol", DVB BlueBook A116, May 2007.

[RFC4301] S. Kent, K. Seo, "Security Architecture for the Internet Protocol", IETF RFC 4301, Dec. 2005.

[RFC4346] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", IETF RFC 4346, April 2006.

[RFC4253] T. Ylonen, C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", IETF RFC 4253, Jan. 2006.

[RFC3819] P. Karn, C. Bormann, G. Fairhurst, D. Grossman, R. Ludwig, J. Mahdavi, G. Montenegro, J. Touch, L. Wood, "Advice for Internet Subnetwork Designers", IETF RFC 3819, July 2004.

[Rijndael]J. Daemen, V. Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard", Springer Verlag, March 2002, pp. 238.

[Subset] D. Naor, M. Naor, J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Proceedings, August 2001, pp. 41-62.

[DVB-CA] "Digital Video Broadcasting (DVB); Support for Use of Scrambling and Conditional Access (CA) within Digital Broadcasting Systems", ETSI ETR 289, Jan. 1996.

[ATSC-CA] "Conditional Access System for Terrestrial Broadcast, Revision A, with Amendment No. 1", Doc. A/70A, Advanced Television Systems Committee, July 2004 (Sept. 2006 for Amendment No. 1).

[DVB-RCS] "Digital Video Broadcasting (DVB); Interaction Channel for Satellite Distribution Systems", ETSI EN 301 790, Sept. 2005.

[RFC4306] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol", IETF RFC 4306, Dec. 2005.

[RFC4046] M. Baugher, R. Canetti, L. Dondeti, F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", IETF RFC 4046, April 2005.

[RFC4535] H. Harney, U. Meth, A. Colegrove, G. Gross, "GSAKMP: Group Secure Association Key Management Protocol", IETF RFC 4535, June 2006.

[RFC3547] M. Baugher, B. Weis, T. Hardjono, H. Harney, "The Group Domain of Interpretation", IETF RFC 3547, July 2003.

[RFC3830] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, K. Norrman, "MIKEY: Multimedia Internet KEYing", IETF RFC 3830, August 2004.

[GKDP] L. Dondetti, J. Xiang, S. Rowles, "GKDP: Group Key Distribution Protocol", IETF draft-ietf-msec-gkdp-01 (expired), March 2006.

[FMKE] L. Duquerroy, S. Josset, "The Flat Multicast Key Exchange protocol", draft-duquer-fmke-01 (expired), Sept. 2004.

[RFC4082] A. Perrig, D. Song, R. Canetti, J. Tygar, B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", IETF RFC 4082, June 2005.

[RFC4082] A. Perrig, D. Song, R. Canetti, J. D. Tygar, B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", IETF RFC 4082, June 2005.

[Arcfour] B. Schneier, "Applied Cryptography Second Edition: protocols algorithms and source in code in C", John Wiley and Sons, New York, 1996.

[Arcfour-Fix] B. Harris, "Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol", IETF RFC 4345, Jan. 2006.

[Standards] B. Burr, "NIST Cryptographic Standards Status Report", PowerPoint presentation, National Institute of Standards and Technology (NIST), April 2006.

[ULEsec-CPI]H. Cruickshank, P. Pillai, S. Iyengar, "Security Extension for Unidirectional Lightweight Encapsulation Protocol", draft-cruickshank-ipdvb-sec-03 (work in progress), July 2007.

[RFC1321] R. Rivest, "The MD5 Message-Digest Algorithm", IETF RFC 1321, April 1992.

[SHA] "The Secure Hash Standard", FIPS 180-2 (+ change notice to include SHA-224), National Institute of Standards and Technology (NIST), August 2002.

[RIPEMD-160]H. Dobbertin, A. Bosselaers, B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD", http://homes.esat.kuleuven.be/~bosselae/ripemd160.html, April 1996.

[RFC2104] H. Krawczyk, M. Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", IETF RFC 2104, Feb. 1997.

[CBCMAC1] M. Bellare, J. Kilian, P. Rogaway, "The security of the cipher block chaining message authentication code", Journal of Computer and System Sciences, Vol. 61, No. 3, Dec. 2000, pp. 362-399.

[CBCMAC2] B. Preneel, P. C. van Oorschot, "MDx-MAC and building fast MACs from hash functions", Proc. Crypto '95: 15th Annual International Cryptology Conference, Santa Barbara, Aug. 1995.

[RFC4418] T. Krovetz, "UMAC: Message Authentication Code using Universal Hashing", IETF RFC 4418, March 2006.

[Poly1305-AES] D. Bernstein, "The Poly1305-AES Message-Authentication Code", Proceedings of Fast Software Encryption, Lecture Notes in Computer Science, Springer-Verlag, 2005.

[XOR-MAC] M. Bellare, R. Guerin, P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions", Advances in Cryptology - Crypto 95 Proceedings, Lecture Notes in Computer Science, Springer-Verlag, 1995.

[PMAC] J. Black, P. Rogaway, "A Block-Cipher Mode of Operation for Parallelizable Message Authentication", Advances in Cryptology - EUROCRYPT '02, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[DSS] "Digital Signature Standard (DSS)", FIPS PUB 186-2 (+ change notice), National Institute of Standards and Technology (NIST), Jan. 2000.

[Order-AE]H. Krawczyk, "The order of encryption and authentication for protecting communications", Proc. Crypto 2001: 21st Annual International Cryptology Conference, Santa Barbara, Aug. 2001.

[IAPM] C. Jutla, "Parallelizable Encryption Mode with Almost Free Message Integrity", NIST proposed mode. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html

[XCBC] V. Gligor, P. Donescu, "Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes", NIST proposed mode, April 2001. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html

[OCB] P. Rogaway, M. Bellare, J. Black, T. Krovetz, "OCB: A Block-Cipher Based Mode of Operation for Efficient Authenticated Encryption", NIST proposed mode, August 2001. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html

[CCM] D. Whiting, R. Housley, N. Ferguson, "AES Encryption & Authentication using CTR Mode & CBC-MAC", IEEE P802.11 802.11-02/001r0, Jan. 2002.

[MD5-Attack]X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", Cryptology ePrint Archive: Report 2004/199, August 2004.

[SHA-1-Attack] X. Wang, Y. Yin, H. Yu, "Finding Collisions in the Full SHA-1", Advances in Cryptology - Crypto 2005 Proceedings, Lecture Notes in Computer Science, Springer-Verlag, Aug. 2005.

[Preimages] J. Kelsey, B. Schneier, "Second Preimages on n-bit Hash Functions for Much Less than 2^n Work", Cryptology ePrint Archive: Report 2004/304, Nov. 2004.

[HMAC2] M. Bellare, "New Proofs for NMAC and HMAC: Security without Collision-Resistance", Advances in Cryptology - Crypto 2006 Proceedings, Lecture Notes in Computer Science Vol. 4117, Springer-Verlag, Sept. 2006.

[Recommendations] "Recommendation for Key Management - Part 1: General (Revised)", SP 800-57, National Institute of Standards and Technology (NIST), May 2006.

Author's Addresses

Michael Noisternig
University of Salzburg
Jakob-Haringer-Str. 2
5020 Salzburg
Austria

Email: mnoist@cosy.sbg.ac.at

Bernhard Collini-Nocker
University of Salzburg
Jakob-Haringer-Str. 2
5020 Salzburg
Austria

Email: bnocker@cosy.sbg.ac.at

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.