PERC Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

J. Mattsson, Ed.
Ericsson
D. McGrew
D. Wing
F. Andreasen
C. Jennings
Cisco

July 8, 2016

Encrypted Key Transport for Secure RTP
draft-ietf-perc-srtp-ekt-diet-01

Abstract

Encrypted Key Transport (EKT) is an extension to Secure Real-time Transport Protocol (SRTP) that provides for the secure transport of SRTP master keys, Rollover Counters, and other information within SRTP. This facility enables SRTP to work for decentralized conferences with minimal control by allowing a common key to be used across multiple endpoints.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Table of Contents

1. Introduction

RTP is designed to allow decentralized groups with minimal control to establish sessions, such as for multimedia conferences. Unfortunately, Secure RTP (SRTP [RFC3711]) cannot be used in many minimal-control scenarios, because it requires that SSRC values and other data be coordinated among all of the participants in a session. For example, if a participant joins a session that is already in progress, that participant needs to be told the SRTP keys (and SSRC, ROC and other details) of the other SRTP sources.

The inability of SRTP to work in the absence of central control was well understood during the design of the protocol; the omission was considered less important than optimizations such as bandwidth conservation. Additionally, in many situations SRTP is used in conjunction with a signaling system that can provide most of the central control needed by SRTP. However, there are several cases in which conventional signaling systems cannot easily provide all of the coordination required. It is also desirable to eliminate the layer violations that occur when signaling systems coordinate certain SRTP parameters, such as SSRC values and ROCs.

This document defines Encrypted Key Transport (EKT) for SRTP and reduces the amount of external signaling control that is needed in a SRTP session that is shared with multiple receivers. EKT securely distributes the SRTP master key and other information for each SRTP source. With this method, SRTP entities are free to choose SSRC values as they see fit, and to start up new SRTP sources (SSRC) with new SRTP master keys (see Section 2.2) within a session without coordinating with other entities via external signaling or other external means.

EKT provides a way for an SRTP session participant, either a sender or receiver, to securely transport its SRTP master key and current SRTP rollover counter to the other participants in the session. This data furnishes the information needed by the receiver to instantiate an SRTP/SRTCP receiver context.

EKT does not control the manner in which the SSRC is generated; it is only concerned with their secure transport.

EKT is not intended to replace external key establishment mechanisms, Instead, it is used in conjunction with those methods, and it relieves them of the burden of tightly coordinating every SRTP source (SSRC) among every SRTP participant.

1.1. Conventions Used In This Document

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

2. Encrypted Key Transport

EKT defines a new method of providing SRTP master keys to an endpoint. In order to convey the ciphertext of the SRTP master key, and other additional information, an additional EKT field is added to SRTP packets. When added to SRTP, the EKT field appears at the end of the SRTP packet, after the authentication tag (if that tag is

present), or after the ciphertext of the encrypted portion of the packet otherwise.

EKT MUST NOT be used in conjunction with SRTP's MKI (Master Key Identifier) or with SRTP's <From, To> [RFC3711], as those SRTP features duplicate some of the functions of EKT.

2.1. EKT Field Formats

The EKT Field uses the format defined below for the Full_EKT_Field and Short_EKT_Field.

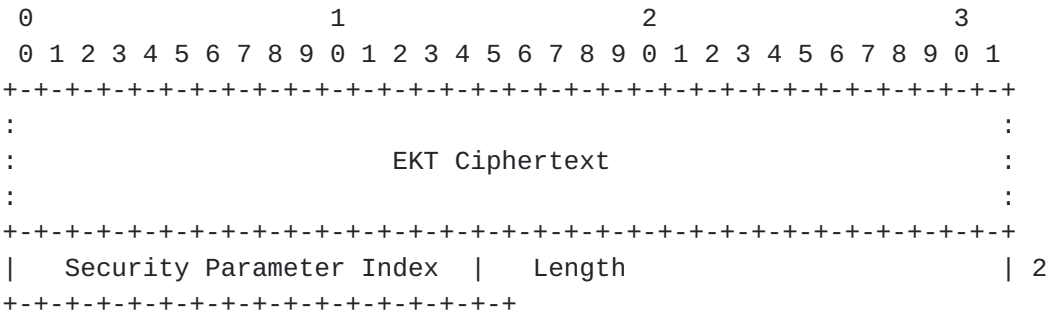

Figure 1: Full EKT Field format

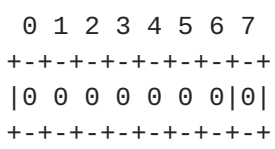

Figure 2: Short EKT Field format

TODO: move following syntax to ABNF. Move name of Short to Empty. Move name of Full to EncryptedKey. Drop extra EKT.

```
EKT_Msg_Type_Full = 2 ;  8 bits
EKT_Msg_Length ; 16 bits. length in octets including type and length

EKT_Plaintext = SRTP_Master_Key || SSRC || ROC || TTL

EKT_Ciphertext = EKT_Encrypt(EKT_Key, EKT_Plaintext)

Full_EKT_Field = EKT_Ciphertext || SPI ||
                           EKT_Msg_Length || EKT_Msg_Type_Full

Short_EKT_Field = '00000000'
```

Figure 3: EKT data formats

These fields and data elements are defined as follows:

EKT_Plaintext: The data that is input to the EKT encryption operation. This data never appears on the wire, and is used only in computations internal to EKT. This is the concatenation of the SRTP Master Key, the SSRC, ROC, and the TTL.

EKT_Ciphertext: The data that is output from the EKT encryption operation, described in Section 2.3. This field is included in SRTP packets when EKT is in use. The length of this field is variable, and is equal to the ciphertext size N defined in Section 2.3. Note that the length of the field is inferable from the SPI field, since the SPI will indicate the cipher being used and thus the size.

SRTP_Master_Key: On the sender side, the SRTP Master Key associated with the indicated SSRC. The length of this field depends on the cipher suite negotiated during call setup for SRTP or SRTCP.

SSRC: On the sender side, this field is the SSRC for this SRTP source. The length of this field is 32 bits.

Rollover Counter (ROC): On the sender side, this field is set to the current value of the SRTP rollover counter in the SRTP context associated with the SSRC in the SRTP or SRTCP packet. The length of this field is 32 bits.

Time to Live (TTL): The maximum amount of time that this key can be used. A unsigned 16 bit integer representing duration in seconds. The SRTP Master key in this message MUST NOT be used for encrypting or decrypting information after this time. Open Issue: does this need to be absolute time not duration? TODO: discuss in security section.

Security Parameter Index (SPI): This field indicates the appropriate EKT Key and other parameters for the receiver to use when processing the packet. Each time a different EKT Key is received, it will have a larger SPI than the previous key (after taking rollover into account). The length of this field is 16 bits. The parameters identified by this field are:

* The EKT cipher used to process the packet.

* The EKT Key used to process the packet.

* The SRTP Master Salt associated with any Master Key encrypted with this EKT Key.

Together, these data elements are called an EKT parameter set. Within each SRTP session, each distinct EKT parameter set that may be used MUST be associated with a distinct SPI value, to avoid ambiguity.

EKT_Msg_Length All EKT message other that Short EKT Field must have a length as the second from last elements. This is the length in octets of the full EKT message including this length field and the following message type.

Message Type The last byte is used to indicate the type of the Field. This MUST be 2 in the Full EKT Field format and 0 in Short EKT Field. Future specifications that define new types SHOULD use even values until all the even code points are consumed to avoid conflicts with pre standards version of EKT that have been deployed. Values less than 64 are mandatory to understand the whole EKT field SHOULD be discarded if it contains message type value that is less than 64 and not implemented.

TODO - add IANA registry for Message Type.

2.2. Packet Processing and State Machine

At any given time, each SRTP/SRTCP source (SSRC) has associated with it a single EKT parameter set. This parameter set is used to process all outbound packets, and is called the outbound parameter set for that SSRC. There may be other EKT parameter sets that are used by other SRTP/SRTCP sources in the same session, including other SRTP/SRTCP sources on the same endpoint (e.g., one endpoint with voice and video might have two EKT parameter sets, or there might be multiple video sources on an endpoint each with their own EKT parameter set). All of the received EKT parameter sets SHOULD be stored by all of the participants in an SRTP session, for use in processing inbound SRTP and SRTCP traffic.

All SRTP master keys MUST NOT be re-used, MUST be randomly generated according to [RFC4086], and MUST NOT be equal to or derived from other SRTP master keys.

Either the Full_EKT_Field or Short_EKT_Field is appended at the tail end of all the SRTP packet.

2.2.1. Outbound Processing

See Section 2.6 which describes when to send an EKT packet with a Full EKT Field. If a Full EKT Field is not being sent, then a Short EKT Field needs to be sent so the receiver can correctly determine how to process the packet.

When an SRTP packet is to be sent with a Full EKT Field, the EKT field for that packet is created as follows, or uses an equivalent set of steps. The creation of the EKT field MUST precede the normal SRTP packet processing.

1. The Security Parameter Index field is set to the value of the Security Parameter Index that is associated with the outbound parameter set.

2. The EKT_Plaintext field is computed from the SRTP Master Key, SSRC, and ROC fields, as shown in Section 2.1. The ROC, SRTP Master Key, and SSRC used in EKT processing SHOULD be the same as the one used in the SRTP processing.

3. The EKT_Ciphertext field is set to the ciphertext created by encrypting the EKT_Plaintext with the EKT cipher, using the EKT Key as the encryption key. The encryption process is detailed in Section 2.3.

4. Then the Full EKT Field is formed using the EKT Ciphertext and the SPI associated with the EKT Key used above. The computed value of the Full EKT Field is written into the packet.

When a packet is sent with the Short EKT Field, the Short EKF Field is simply appended to the packet.

2.2.2. Inbound Processing

Inbound EKT processing MUST take place prior to the usual SRTP or SRTCP processing. The following steps show processing as packets are received in order.

1. The final byte is checked to determine which EKT format is in use. When an SRTP or SRTCP packet contains a Short EKT Field,

the Short EKT Field is removed from the packet then normal SRTP or SRTCP processing occurs. If the packet contains a Full EKT Field, then processing continues as described below.

2. The combination of the SSRC and the Security Parameter Index (SPI) field is used to find which EKT parameter set should be used when processing the packet. If there is no matching SPI, then the verification function MUST return an indication of authentication failure, and the steps described below are not performed. EKT parameter set contains the EKT Key, EKT Cipher, and SRTP Master Salt.

3. The EKT Ciphertext authentication is checked and it is decrypted, as described in Section 2.3, using the EKT Key and EKT Cipher found in the previous step. If the EKT decryption operation returns an authentication failure, then the packet processing stops.

4. The resulting EKT Plaintext is parsed as described in Section 2.1, to recover the SRTP Master Key, SSRC, and ROC fields. The Master Salt that is assocted with the EKT Keys used to do the decription is also retreived.

5. The SRTP Master Key, ROC, and Master Salt from the prevous step are saved in a map indexed by the SSRC found in the EKT Plaintext and can be used for any future crypto operations for inbound or outbound packets with the that SSRC. Outbound packets SHOULD continue to use the old key for 250 ms after receipt of the new key. This gives all the receivers in the system time to get the new key before they start receiving media encrypted with the new key. The key MUST NOT be used beyond the lifetime found in the TTL field.

6. At this point, EKT processing has successfully completed, and the normal SRTP or SRTCP processing takes place including replay protection.

Implementation note: the receiver may want to have a sliding window to retain old SRTP master keys (and related context) for some brief period of time, so that out of order packets can be processed as well as packets sent during the time keys are changing.

2.3. Ciphers

EKT uses an authenticated cipher to encrypt and authenticate the EKT Plaintext. We first specify the interface to the cipher, in order to abstract the interface away from the details of that function. We then define the cipher that is used in EKT by default. The default

cipher described in Section 2.3.1 MUST be implemented, but another cipher that conforms to this interface MAY be used, in which case its use MUST be coordinated by external means (e.g., key management).

An EKT cipher consists of an encryption function and a decryption function. The encryption function E(K, P) takes the following inputs:

o a secret key K with a length of L bytes, and

o a plaintext value P with a length of M bytes.

The encryption function returns a ciphertext value C whose length is N bytes, where N is at least M. The decryption function D(K, C) takes the following inputs:

o a secret key K with a length of L bytes, and

o a ciphertext value C with a length of N bytes.

The decryption function returns a plaintext value P that is M bytes long, or returns an indication that the decryption operation failed because the ciphertext was invalid (i.e. it was not generated by the encryption of plaintext with the key K).

These functions have the property that D(K, E(K, P)) = P for all values of K and P. Each cipher also has a limit T on the number of times that it can be used with any fixed key value. For each key, the encryption function MUST NOT be invoked on more than T distinct values of P, and the decryption function MUST NOT be invoked on more than T distinct values of C.

Security requirements for EKT ciphers are discussed in Section 5.

2.3.1. The Default Cipher

The default EKT Cipher is the Advanced Encryption Standard (AES) Key Wrap with Padding [RFC5649] algorithm. It requires a plaintext length M that is at least one octet, and it returns a ciphertext with a length of N = M + 8 octets. It can be used with key sizes of L = 16, and 32 octets, and its use with those key sizes is indicated as AESKW_128, or AESKW_256, respectively. The key size determines the length of the AES key used by the Key Wrap algorithm. With this cipher, T=2^48.

length of SRTP transform	length of EKT transform	EKT plaintext	EKT ciphertext	length of Full EKT Field
AES-128	AESKW_128	26	40	42
AES-256	AESKW_256	42	56	58

Figure 4: AESKW Table

As AES-128 is the mandatory to implement transform in SRTP [RFC3711], AESKW_128 MUST be implemented for EKT.

For all the SRTP transforms listed in the table, the corresponding EKT transform MUST be used, unless a stronger EKT transform is negotiated by key management.

2.3.2. Other EKT Ciphers

Other specifications may extend this one by defining other EKT ciphers per Section 7. This section defines how those ciphers interact with this specification.

An EKT cipher determines how the EKT Ciphertext field is written, and how it is processed when it is read. This field is opaque to the other aspects of EKT processing. EKT ciphers are free to use this field in any way, but they SHOULD NOT use other EKT or SRTP fields as an input. The values of the parameters L, M, N, and T MUST be defined by each EKT cipher, and those values MUST be inferable from the EKT parameter set.

2.4. Synchronizing Operation

If a source has its EKT key changed by the key management, it MUST also change its SRTP master key, which will cause it to send out a new Full EKT Field. This ensures that if key management thought the EKT key needs changing (due to a participant leaving or joining) and communicated that in key management to a source, the source will also change its SRTP master key, so that traffic can be decrypted only by those who know the current EKT key.

2.5. Transport

EKT SHOULD be used over SRTP, and other specification MAY define how to use it over SRTCP. SRTP is preferred because it shares fate with transmitted media, because SRTP rekeying can occur without concern for RTCP transmission limits, and to avoid SRTCP compound packets with RTP translators and mixers.

2.6. Timing and Reliability Consideration

A system using EKT learns the SRTP master keys distributed with Full EKT Fields send with the SRTP, rather than with call signaling. A receiver can immediately decrypt an SRTP provided the SRTP packet contains a Full EKT Field.

This section describes how to reliably and expediently deliver new SRTP master keys to receivers.

There are three cases to consider. The first case is a new sender joining a session which needs to communicate its SRTP master key to all the receivers. The second case is a sender changing its SRTP master key which needs to be communicated to all the receivers. The third case is a new receiver joining a session already in progress which needs to know the sender's SRTP master key.

New sender: A new sender SHOULD send a packet containing the Full EKT Field as soon as possible, always before or coincident with sending its initial SRTP packet. To accommodate packet loss, it is RECOMMENDED that three consecutive packets contain the Full EKT Field be transmitted.

Rekey: By sending EKT over SRTP, the rekeying event shares fate with the SRTP packets protected with that new SRTP master key.

New receiver: When a new receiver joins a session it does not need to communicate its sending SRTP master key (because it is a receiver). When a new receiver joins a session the sender is generally unaware of the receiver joining the session. Thus, senders SHOULD periodically transmit the Full EKT Field. That interval depends on how frequently new receivers join the session, the acceptable delay before those receivers can start processing SRTP packets, and the acceptable overhead of sending the Full EKT Field. The RECOMMENDED frequency is the same as the key frame frequency if sending video and every 100ms for audio.

3. Use of EKT with DTLS-SRTP

This document defines an extension to DTLS-SRTP called Key Transport. The EKT with the DTLS-SRTP Key Transport enables secure transport of EKT keying material from one DTLS-SRTP peer to another. This enables those peers to process EKT keying material in SRTP (or SRTCP) and retrieve the embedded SRTP keying material. This combination of protocols is valuable because it combines the advantages of DTLS (strong authentication of the endpoint and flexibility) with the advantages of EKT (allowing secure multiparty RTP with loose coordination and efficient communication of per-source keys).

3.1. DTLS-SRTP Recap

DTLS-SRTP [RFC5764] uses an extended DTLS exchange between two peers to exchange keying material, algorithms, and parameters for SRTP. The SRTP flow operates over the same transport as the DTLS-SRTP exchange (i.e., the same 5-tuple). DTLS-SRTP combines the performance and encryption flexibility benefits of SRTP with the flexibility and convenience of DTLS-integrated key and association management. DTLS-SRTP can be viewed in two equivalent ways: as a new key management method for SRTP, and a new RTP-specific data format for DTLS.

3.2. EKT Extensions to DTLS-SRTP

This document adds a new TLS negotiated extension called "ekt". This adds a new TLS content type, EKT, and a new negotiated extension EKT. The DTLS server includes "ekt" in its TLS ServerHello message. If a DTLS client includes "ekt" in its ClientHello, but does not receive "ekt" in the ServerHello, the DTLS client MUST NOT send DTLS packets with the "ekt" content-type.

Using the syntax described in DTLS [RFC6347], the following structures are used:

```
enum {
  ekt_key(0),
  ekt_key_ack(1),
  ekt_key_error(254),
  (255)
} SRTPKeyTransportType;

struct {
  SRTPKeyTransportType keytrans_type;
  uint24 length;
  uint16 message_seq;
  uint24 fragment_offset;
  uint24 fragment_length;
  select (SRTPKeyTransportType) {
     case ekt_key:
        EKTkey;
   };
} KeyTransport;

enum {
 RESERVED(0),
 AESKW_128(1),
 AESKW_256(3),
} ektcipher;

struct {
  ektcipher EKT_Cipher;
  uint EKT_Key_Value<1..256>;
  uint EKT_Master_Salt<1..256>;
  uint16 EKT_SPI;
} EKTkey;
```

Figure 5: Additional TLS Data Structures

The diagram below shows a message flow of DTLS client and DTLS server using the DTLS-SRTP Key Transport extension.

Figure 6: Handshake Message Flow

3.3. Offer/Answer Considerations

When using EKT with DTLS-SRTP, the negotiation to use EKT is done at the DTLS handshake level and does not change the [RFC3264] Offer / Answer messaging.

4. Sending the DTLS EKT_Key Reliably

The DTLS ekt_key is sent using the retransmiions specified in Section 4.2.4. of DTLS [RFC6347].

5. Security Considerations

EKT inherits the security properties of the DTLS-SRTP (or other) keying it uses.

With EKT, each SRTP sender and receiver MUST generate distinct SRTP master keys. This property avoids any security concern over the re-use of keys, by empowering the SRTP layer to create keys on demand. Note that the inputs of EKT are the same as for SRTP with key-

sharing: a single key is provided to protect an entire SRTP session. However, EKT remains secure even when SSRC values collide.

The EKT Cipher includes its own authentication/integrity check. For an attacker to successfully forge a full EKT packet, it would need to defeat the authentication mechanisms of the EKT Cipher authentication mechanism.

The presence of the SSRC in the EKT_Plaintext ensures that an attacker cannot substitute an EKT_Ciphertext from one SRTP stream into another SRTP stream.

An attacker who tampers with the bits in Full_EKT_Field can prevent the intended receiver of that packet from being able to decrypt it. This is a minor denial of service vulnerability.

An attacker could send packets containing a Full EKT Field, in an attempt to consume additional CPU resources of the receiving system by causing the receiving system will decrypt the EKT ciphertext and detect an authentication failure

EKT can rekey an SRTP stream until the SRTP rollover counter (ROC) needs to roll over. EKT does not extend SRTP's rollover counter (ROC), and like SRTP itself EKT cannot properly handle a ROC rollover. Thus even if using EKT, new (master or session) keys need to be established after 2^48 packets are transmitted in a single SRTP stream as described in Section 3.3.1 of [RFC3711]. Due to the relatively low packet rates of typical RTP sessions, this is not expected to be a burden.

The confidentiality, integrity, and authentication of the EKT cipher MUST be at least as strong as the SRTP cipher.

Part of the EKT_Plaintext is known, or easily guessable to an attacker. Thus, the EKT Cipher MUST resist known plaintext attacks. In practice, this requirement does not impose any restrictions on our choices, since the ciphers in use provide high security even when much plaintext is known.

An EKT cipher MUST resist attacks in which both ciphertexts and plaintexts can be adaptively chosen and adversaries that can query both the encryption and decryption functions adaptively.

6. Open Issues

What length should the SPI be?

Should we limit the number of saved SPI for a given SSRC? Or limit the lifetime of old ones after a new one is received? At some level this may not matter because even if the a SRTP packet is injected with an old value, it will be discards by the RTP stack for being old. It is more important that new things are encrypted with the most recent EKT Key.

How many bits to differentiate different types of packets and allow for extensibility?

Given the amount of old EKT deployed, should the Full EKT use a a different code point than the "1" at the end?

Do we need AES-192?

7. IANA Considerations

No IANA actions are required.

8. Acknowledgements

Thanks to David Benham, Eddy Lem, Felix Wyss, Jonathan Lennox, Kai Fischer, Lakshminath Dondeti, Magnus Westerlund, Michael Peck, Nermeen Ismail, Paul Jones, Rob Raymond, and Yi Cheng for fruitful discussions, comments, and contributions to this document.

This draft is a cut down version of draft-ietf-avtcore-srtp-ekt-03 and most of the text here came from that draft.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <http://www.rfc-editor.org/info/rfc5649>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <http://www.rfc-editor.org/info/rfc5764>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

9.2. Informative References

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

Authors' Addresses

John Mattsson (editor)
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 43 501
Email: john.mattsson@ericsson.com

David A. McGrew
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
US

Phone: (408) 525 8651
Email: mcgrew@cisco.com
URI: http://www.mindspring.com/~dmcgrew/dam.htm

Dan Wing
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
US

Phone: (408) 853 4197
Email: dwing@cisco.com

Flemming Andreason
Cisco Systems
499 Thornall Street
Edison, NJ 08837
US

Email: fandreas@cisco.com

Cullen Jennings
Cisco Systems
Calgary, AB
Canada

Email: fluffy@iii.ca