                        IPSec Re-keying Issues
                   <draft-jenkins-ipsec-rekeying-00.txt>

Status of this Memo

   Informational

   This memo provides information for the Internet community. This memo
   does not specify an Internet standard of any kind.

   Distribution of this memo is unlimited.

Copyright Notice

Table of Contents

Revision History

  September 23, 1998   Initial Release

[1](#). Introduction

   For a number of reasons, re-keying in IPSec has become problematic,
   such that packets can get dropped by IPSec implementations during
   re-keying. Worse, there exists the possibility that IPSec
   implementations from different vendors may not be interoperable
   because of the way they re-key.

   The purpose of this paper is to propose methods of performing both
   phase 1 and phase 2 re-keying for IPSec implementations in such a
   way to to minimize packet loss and to maximize compatibility.

   The initial focus in on phase 2 re-keying; it is then extended to
   phase 1 re-keying. The need for this document in each case is
   initially discussed, followed by a recommendation for re-keying
   within the protocol framework established by version 1.0 of the
   IPSec documents.

   Finally, recommendations for IPSecond are made to best solve the re-
   keying problems in a manner that is not possible within the
   constraints of the existing IPSec documents.

[2](#). Phase 2 SA Re-keying

   This section discusses phase 2 re-keying issues and makes
   recommendations to minimize the impact of these issues.

[2.1](#) Phase 2 Re-keying Issues

   The issues associated with phase 2 re-keying listed below. Some of
   the points are expanded upon later.

   1) There is no specification how re-keying is to be done.

   2) The existing drafts appear contradictory in their
       recommendations on the usage of multiple phase 2 SAs.

   3) Some recent implementations have shipped with a method of re-
       keying that will not perform reliably under real world network
       conditions.

   4) The use of the Delete notification is not required.

   5) A variety of re-keying behaviours have been observed, some of
       which are incompatible.

6) The commit bit is not yet widely implemented, and its use as
    described is confusing. Further, while the documentation
    requires its support, its use is not required.

7) A race condition exists at SA set up, exacerbating re-keying
    issues.

2.1.1 **Inconsistent SA Use Recommendation**

The issue of inconsistent SA usage recommendations is examined
further here.

From paragraph 2 of Section 9 of [IKE]:

  An implementation may wish to negotiate a range of SAs when
  performing Quick Mode. By doing this they can speed up the "re-
  keying". Quick Mode defines how KEYMAT is defined for a range of
  SAs. When one peer feels it is time to change SAs they simply use
  the next one within the stated range. A range of SAs can be
  established by negotiating multiple Sas (identical attributes,
  different SPIs) with one Quick Mode.

While the document does not define what "... the next one ..."
means, this paragraph strongly implies that there is no required
order for the use of phase 2 SAs that have been negotiated in a
phase 1 SA, or that multiple SAs may be pre-negotiated and used at
will.

However, this appears to be contradicted by paragraph 3 of section
4.3 of [ISAKMP]:

  Modification of a Protocol SA (phase 2 negotiation) follows the
  same procedure as creation of a Protocol SA. The creation of a new
  SA is protected by the existing ISAKMP SA. There is no
  relationship between the two Protocol SAs. A protocol
  implementation SHOULD begin using the newly created SA for
  outbound traffic and SHOULD continue to support incoming traffic
  on the old SA until it is deleted or until traffic is received
  under the protection of the newly created SA. As stated previously
  in this section, deletion of an old SA is then dependent on local
  security policy.

Many implementations have interpreted this to mean that the new SA
should be used for outbound in preference to the old SA. In other
words, the old SA should be abandoned as soon as possible.

[2.1.2](#) Observed Behaviours

The following behaviours have been observed by various vendors'
implementations when devices have set up a second phase 2 SA.

1) The device continues to use the old SA until it naturally
    expires, then switches to the new SA.

2) The device immediately begins using the new SA.

3) The device immediately drops the old SA.

4) The device never sends a Delete notification.

5) The device always sends a Delete notification.

6) The device deletes the old SA some time after re-keying, but
    before the end of its natural lifetime.

7) A device wants to keep more than one SA up all the time.

All of these behaviours are permitted under the current documents.
However, even when phase 2 exchange packets are not lost, it can be
seen that interoperability is not always possible due the
combinations of behaviours listed above.

[2.1.3](#) SA Set-up Race Condition

Further, behaviour 2 above is not a good behaviour, as illustrated
below. In this example, the initiator is a gateway capable of
handling full T3 bandwidth rates, while the responder is a PC
running a software IPSec implementation, and it is overloaded.

In the illustration, QM1 refers to the first quick mode message, QM2
to the second quick mode message and QM3 to the third.

```
              Initiator                         Responder

    QM1 sent    ----
                    -------
                        ------------
                                    ---------------> QM1 received
                                                 |
                                                 |
                                                 | QM1 processed
                                                 |
                                                 |
                                    --------------- QM2 sent
                        ------------
                    -------
    QM2 rec.   <----
    process    |
    QM3 sent   -----
               *     -------
  packet on new SA        ------------
                _____                  ---------------> QM3 received
                    _____                          |
                        _____                | QM3 processing
                                    _____|
                                                 | packet dropped
                                                 |
                                                 * new SA set up
```
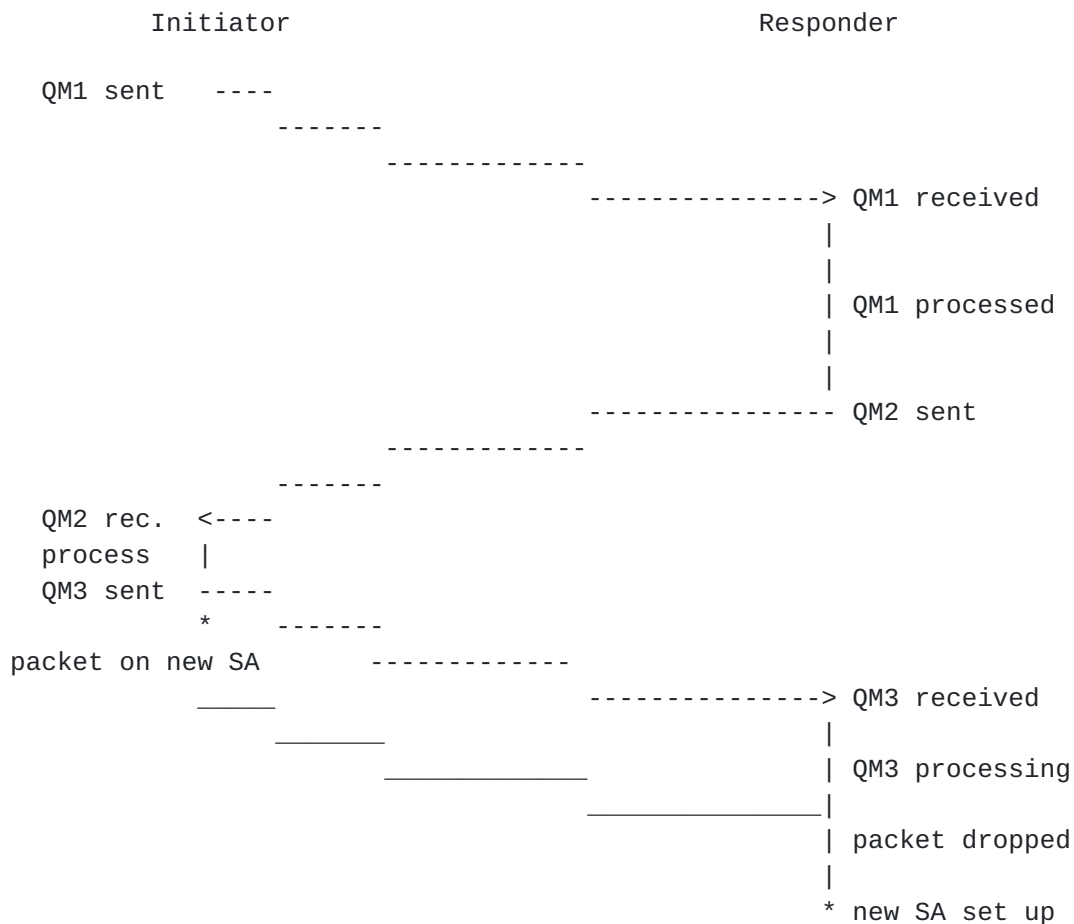
                Figure 2-1 Race Condition Sequence Chart

   By the time the responder has set up the new SA, packets protected
   by that SA have already started arriving from the initiator. This
   causes them to be dropped by the responder. This case is further
   complicated by the possibility of packets taking different paths
   through the network, so theoretically, the third quick mode message
   could arrive after packets protected by the new SA.

   Additionally, since all IKE packets are based on UDP, there is no
   guarantee that QM3 even arrives at the peer, so making assumptions
   about new SA use based on the transmission time of a packet will
   still lead to failures in the field.

   To reduce the effects of packet loss, some implementations were
   observed to blindly transmit QM3 multiple times, back to back.

   This can reduce the probability that the peer does not get QM3, but
   cannot eliminate it.

If the behaviour of the initiator was to delay usage of the new SA
for outbound traffic, this would cause failures for those
implementations that immediately delete the old SA. Therefore, the
behaviour of delaying use of the new SA and immediately deleting the
old SA are incompatible.

2.1.4 Commit Bit Interaction

The use of the commit bit can solve the race condition illustrated
in the previous section when asserted by the responder during quick
mode. However, it suffers from the following problems:

1) Use of the commit bit is not well defined. The present
    documentation specifies its used for phase 1 and phase 2, but
    mentions phase 2 specific details. There are also issues
    related to how the subsequent Connected notification fits in
    with the quick mode exchange.

2) While its support is required, its use is not. Current
    indications are that its use is not widespread.

3) Its use may make implementations susceptible to a denial of
    service attack by forcing initiators to wait for a Connected
    notification that may never come. While this is only one of
    many very basic possible denial of service attacks on IKE, this
    is not an excuse to leave the existing implementation as it is.

2.2 Solution Examination

This section details the operation of some possible behaviours, with
the intent of arriving at a best possible phase 2 re-keying
mechanism under the constraints of the existing documents.

In all the examples, the term "sets up a new outbound SA" means that
the new outbound SA will be chosen in favour of the old one. Whether
the SA is actually created before that time or not is implementation
dependent.

2.2.1 Responder Pre-Set Up

As a starting point, the responder pre-set up method of re-keying is
examined. Note that it will work with most of the behaviours
observed in the field.

   In this method, SAs are treated separately as inbound and outbound,
   as well as old and new. Further, it takes advantage of the fact that
   the responder knows what the SA is going to be after the second
   quick mode message is sent.

   Implicit acknowledgement of the reception of the third quick mode
   message by the responder is provided by use of the new SA in the
   outbound direction. The initiator should not use the new outbound SA
   before that time.

   Additionally, it does not require use of the Delete notification.
   This is important since, even if it is always sent, it is an
   unacknowledged UDP packet and can be lost.

2.2.1.1 **Normal Conditions**

   The following is the operation under normal (successful) conditions.

```
                 Initiator                  Responder

             Inbound    Outbound       Inbound    Outbound
               |           |              |           |
      1     ----------------              |           |
               |           | -----------  |           |
               |           |              --------------------->  2
               |           |              |       |     |
               |           |              --------------------   3
               |           | -----------  |  *4   |
      5     <---------------              |  |     |
             | |           |              |  |     |
      6     ----------------              |  |     |
               |  *7       | -----------  |  |     |
               |  |        |              --------------------> 8
               |  |        |              |  |    |       |
               |  |        |              |  |    |       *
               |  |        |              |  |    |  *9
               |  |        |              |  |   *10 |
               |  |        |              |  |       |
               |  *11      |              |  |       |
               |  |        |  *12         |  |       |
               |  |      *13 |            |  |       |
             *14|            |            |  |       |
               |            |            |  *15      |
               |            |            *16|        |
               |            |              |        |
```
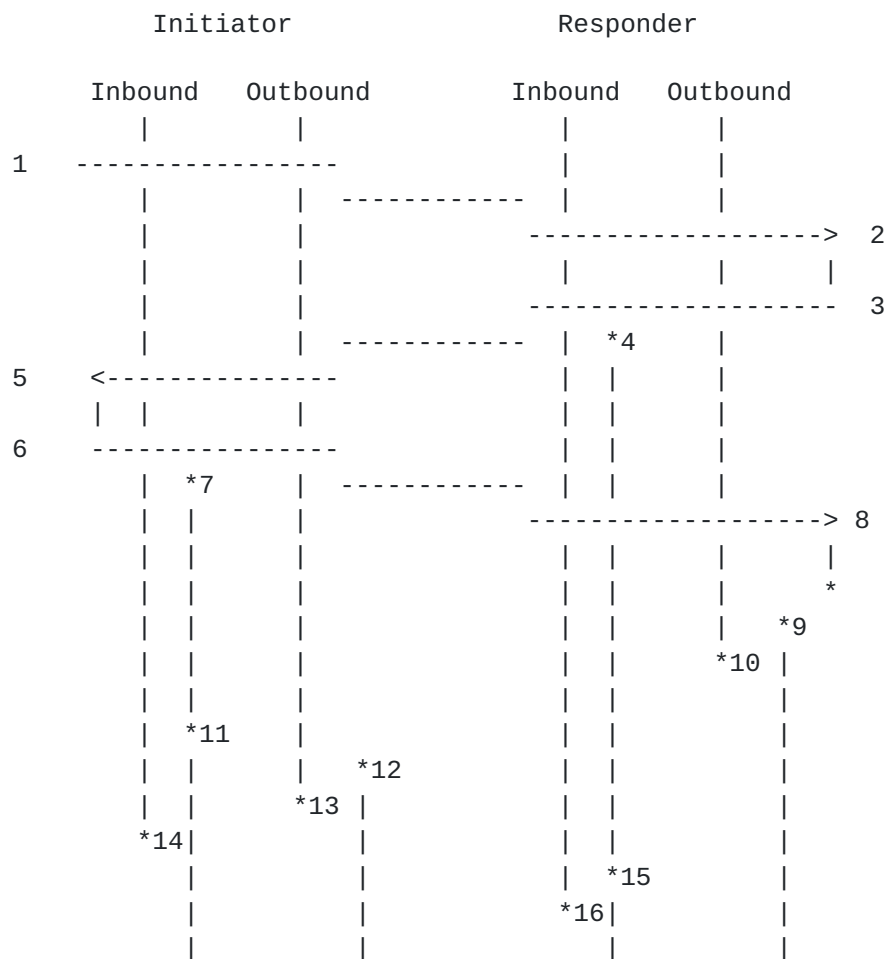
                 Figure 2-2 SA Pre-Set Up Sequence Chart

Events

1) Initiator sends first quick mode message.

2) Responder receives first quick mode message.

3) Responder sends second quick mode message.

4) Responder sets up new inbound SA. This is to handle the case
    where the initiator starts transmitting on the new SA
    immediately after sending the third quick mode message.

5) Initiator receives second quick mode message.

6) Initiator sends third quick mode message.

7) Initiator sets up new inbound SA.

8) Responder receives third quick mode message.

9) Responder sets up new outbound SA.

10) Responder deletes old outbound SA.

11) Traffic from responder to initiator arrives at initiator on new
    SA.

12) Initiator sets up new outbound SA.

13) Initiator deletes old outbound SA.

14) Initiator deletes old inbound SA.

15) Traffic from initiator to responder arrives at responder on new
    SA.

16) Responder deletes old inbound SA.

While appearing complicated, it enables the lossless transfer from
one SA to another while supporting almost all other behaviours.

Support for and use of the Delete notification is unchanged.

2.2.1.2 **Dropped Packet Conditions**

In this case, the event list is modified to show what happens when each packet is dropped once. The event numbers refer to those illustrated in Figure 2.

1) Initiator sends first quick mode message.

e) Packet is dropped during transmission.

1b) Initiator times out waiting for second quick mode message.

1) Initiator re-sends first quick mode message.

2) Responder receives first quick mode message.

3) Responder sends second quick mode message.

4) Responder sets up new inbound SA. This is to handle the case where the initiator starts transmitting on the new SA immediately after sending the third quick mode message.

e) Packet is dropped during transmission.

1b) or 7b) Responder times out waiting for third quick mode message.

1) or 3) Responder re-sends second quick mode message.

5) Initiator receives second quick mode message.

6) Initiator sends third quick mode message.

7) Initiator sets up new inbound SA.

e) Packet is dropped during transmission.

7b) Responder times out waiting for third quick mode message.

3) Responder re-sends second quick mode message.

5) Initiator receives second quick mode message again.

6) Initiator re-sends third quick mode message.

8) Responder receives third quick mode message.

and so on, as for normal operation.

2.2.1.3 Failed Negotiation

In this case, the second quick mode packet has an invalid hash, and
the initiator sends the notification to the peer. Again, the event
numbers refer to those illustrated in Figure 2.

1) Initiator sends first quick mode message.

2) Responder receives first quick mode message.

3) Responder sends second quick mode message.

4) Responder sets up new inbound SA. This is to handle the case
    where the initiator starts transmitting on the new SA
    immediately after sending the third quick mode message.

5) Initiator receives second quick mode message.

e) Hash (or other parameter) fails.

e1) Initiator sends notification to responder.

e2) Responder receives notification.

e3) Responder deletes new inbound SA.

A similar operation would occur if retry counters expire for packet
re-transmissions.

2.2.1.4 Responder Pre-Set Up Security Hole

In the failed negotiation case, the need to delete the invalid
inbound SA raises the issue of a temporary hole, in that the
responder allows inbound packets while waiting for the third quick
mode message. However, if the inbound SA is not set up ahead of
time, initiators that immediately transmit on the new outbound SA
will cause packets to be dropped.

It also illustrates why the proposal above made the usage of the
outbound SA by the initiator wait until there is an indication of
the use of the SA by the responder.

2.2.2 Recommended Re-keying Method

In this method, the previous method is modified to remove the risk
of the security hole. It also simplifies the operation somewhat, but

at the expense of lost packets if the initiator's behaviour is such
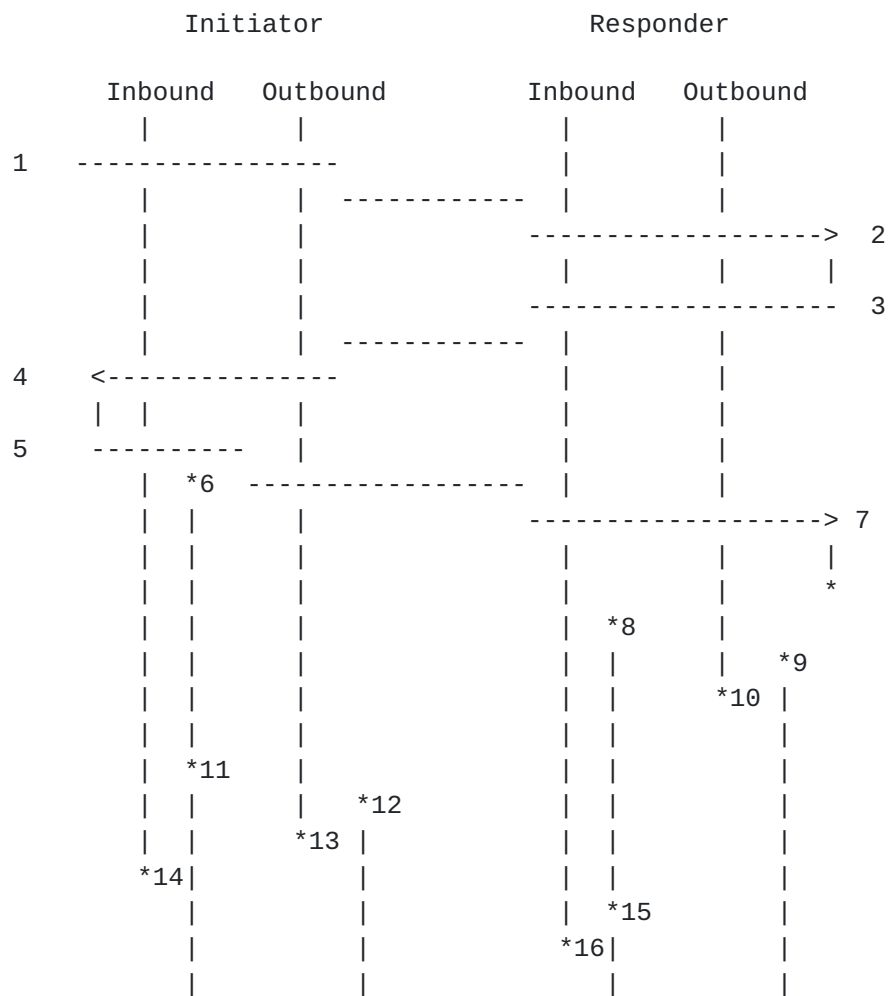that it immediately uses the new SA for its outbound traffic.

```
              Initiator                    Responder

          Inbound    Outbound       Inbound    Outbound
             |           |             |           |
   1    -----------------             |           |
             |           |  -----------   |           |
             |           |      ------------------->  2
             |           |             |       |    |
             |           |      -------------------  3
             |           |  -----------   |           |
   4      <---------------             |           |
          |  |           |             |           |
   5      ----------     |             |           |
          |   *6  ------------------   |           |
          |  |       |      ------------------->  7
          |  |       |             |       |    |
          |  |       |             |       |    *
          |  |       |          |  *8    |
          |  |       |          |  |    |   *9
          |  |       |          |  |   *10 |
          |  |       |          |  |        |
          |  *11     |          |  |        |
          |  |       |  *12     |  |        |
          |  |     *13 |        |  |        |
         *14|          |        |  |        |
             |          |        |  *15     |
             |          |      *16|         |
             |          |        |          |
```

       Figure 2-3 Recommended Phase 2 Re-key Sequence Chart

   1)   Initiator sends first quick mode message.

   2)   Responder receives first quick mode message.

   3)   Responder sends second quick mode message.

   4)   Initiator receives second quick mode message.

   5)   Initiator sends third quick mode message.

   6)   Initiator sets up new inbound SA.

   7)   Responder receives third quick mode message.

8)   Responder set up new inbound SA.

9)   Responder sets up new outbound SA.

10) Responder deletes old outbound SA.

11) Traffic from responder to initiator arrives at initiator on new
     SA.

12) Initiator sets up new outbound SA.

13) Initiator deletes old outbound SA.

14) Initiator deletes old inbound SA.

15) Traffic from initiator to responder arrives at responder on new
     SA.

16) Responder deletes old inbound SA.

Note that deletion of the old inbound SA by the initiator could be
further delayed if protection against loss of packets on the old SA
from different and slower network paths is desired.

2.2.2.1 **Dropped Quick Mode 3 Message**

In cases where the third quick mode message is dropped, the
responder must request re-transmission of it by re-sending the
second quick mode message. The existence of traffic on the new
inbound SA at the initiator should not be used as an implicit
acknowledgement for the following reasons:

1)   There may be no traffic for the responder to send.

2)   The responder may be implemented to use the old SA until its
     natural expiration.

2.2.2.2 **Absence of Traffic**

The proposed implementation uses the presence of traffic from the
responder on new SAs to provide an implied acknowledgement for the
purposes of switching to the new SA. However, if there is no traffic
from the responder, the implied acknowledgement will not appear.

A similar behaviour is exhibited by implementations that continue to
use old SAs until their natural expiration.

However, due to the number of implementations that delete old SAs 30
seconds after negotiating a new one, the same behaviour has the best
chance of interoperability, and of not dropping packets when traffic
does restart.

Therefore, it is recommended that implementations delete old SAs and
start using new SAs 30 seconds after negotiating new SAs. Use of the
Delete notification is strongly recommended in cases where the peer
implementation is continuing to use the old SA.

2.2.2.3 Compatibility With Observed Behaviours

When operating with behaviours that use the new SA immediately, this
method performs equivalently when this method is used by the
responder. When used by the initiator, the performance will depend
on when the responder deletes the old inbound SA.

When operating with behaviours that continue to use the old SA, this
method performs as described in the dropped quick mode three example
above when used by the initiator. When used by the responder, there
is no change in operation, since the responder will wait until the
new SA is used before deleting the old SA.

However, as stated in a previous section, it is recommended that the
initiator keep the old SA (both inbound and outbound) for only 30
seconds after creation of the new SA in cases where traffic is not
detected on the new SA.

2.2.2.4 Compatibility with Commit Bit

As stated earlier, use of the commit bit as described in the drafts
is confusing.

For the purposes of this document, its use is interpreted to mean
the following:

   "I have set the commit bit. Do not use the SA created by this
   negotiation until I send you the Connected notification."

In other words, the purpose of the commit is to delay a peer's usage
of its outbound SA until it has received the Connected notification.

While sounding simple, this suffers from some of the same problems
as the negotiation without the commit bit. When used as part of a
quick mode negotiation, the effect is that the Connected
notification is now similar to the third quick mode message with the

roles of the initiator and responder reversed (or not reversed if set by the initiator).

Specifically, the Connected notification can still be dropped. This will result in the intended receiver of the Connected notification never sending on the new SA. Also, if the intended receiver of the Connected notification does not set up the new SA until receiving the Connected notification, the same race condition exists if the sender of the notification starts using the new outbound SA immediately after sending the notification.

This problem is further exacerbated by the lack of tight integration of the Connected notification with quick mode. In other words, it may not be possible to request re-transmission of the Connected notification by re-sending the third quick mode message.

The impact of these effects can be eliminated by the following rules:

1) The initiator should set up its inbound SA immediately after sending the third quick mode message regardless of the state of the commit bit.

2) Traffic sense on the initiator's new inbound SA should trigger the use of the new outbound SA to detect cases when the Connected notification is dropped.

The recommended proposal does not allow built-in support of the commit bit. It does allow responders that use the commit bit to detect reception of the Connected notification by the initiator due to the presence of traffic on the inbound SA. However, this works only if there is traffic, so it cannot be considered a usefull method to perform this function.

The recommended proposal does cause the initiator to delay usage of a new SA until it is set up. This is the primary use of the commit bit, so use of this proposal makes the use of the commit bit unnecessary except for the setting up of the first phase 2 SA.

However, other uses of the commit bit or its equivalent function may appear, such as delaying of SA use in key recovery implementations. In these cases, the re-keying method proposed here does not interfere with commit bit usage.

2.2.2.5 **Implementation Notes**

The presence of traffic on the new SA can be part of the expiration
checking operation, and does not need to occur instantaneously,
although it must occur before the 30 second no traffic SA deletion
criteria. As long as the new SA is negotiated with enough time
before the expiration of the old one, the detection of traffic on
the new SA can be on the order of seconds with no ill effects.

Since SAs will likely have traffic counters anyway, this method
requires only the addition of a flag that indicates it is a new SA.
When the expiration process checks for aging and expired SAs, it can
also check for new SAs with a non-zero traffic count. When detected,
the SA is marked as non-new, and the remaining operations can be
performed.

2.3 **Conclusions**

The final re-keying method is the best compromise between security
and interoperability within the framework of the current IPSec
documents.

3. **Phase 1 Re-keying**

This section makes a proposal for main mode re-keying. This proposal
is necessary for many of the same reasons a phase 2 re-keying
proposal is necessary.

1) The rules for phase 1 re-keying are not specified in the
    drafts.

2) Adhoc implementations have lead to poor implementations and
    possible interoperability issues.

The goal of the proposed phase 1 re-keying method is to provide
secure, lossless communications. This means that there should be no
dropped traffic during re-keying, but also that there should be no
further traffic if re-keying fails.

3.1 **Phase 1 Re-keying Requirements**

The two reasons for re-keying a phase 1 SA are for freshness (time
or traffic) of the phase 1 keying material (affecting its ability to
protect phase 2 negotiations) and for re-authentication of the
encrypting devices.

This implies that there is no inherent need to delete other SAs
created by an expired phase 1 SA as long as an immediate attempt to
create a new phase 1 SA is made to verify authentication. If this
fails, then the SAs created within previous phase 1 SAs must be
deleted. This provides the authentication protection of the original
phase 1 SA. Note that this does not preclude any requirements for
early termination of SAs due to certificate revocation, for example.

However, the automatic re-keying of phase 1 SAs means that SAs could
live independent of traffic, since re-keying of both phase 1 and
phase 2 SAs takes place with no traffic triggers. In other words,
SAs that are no longer necessary may never disappear. If an
implementation waits until traffic starts using pre-existing phase 2
SAs before re-keying a phase 1 SA, that traffic could be allowed to
pass unauthenticated for the time that it takes to negotiate. The
difference between this case and the case of immediately
renegotiating is that the traffic could be flowing at some arbitrary
time after the phase 1 SA has expired (but before the phase 2 SA has
expired) and outside the authenticated time, while in the other
case, re-authentication of the SAs effectively happens at the end of
their authenticated lifetime.

This suggests that a traffic monitoring capability should be part of
implementations that need to delete idle or unused SAs. As such, it
is not given further consideration in this document, since it is
beyond the scope of this document.

A further implication of not deleting the phase 2 SAs is that there
is no need to overlap phase 1 SAs. That is, the second phase 1 SA
can be negotiated after the first phase 1 SA expires with no loss of
traffic since the phase 2 SA is still in place.

(There may be issues of simultaneous expiration of phase 1 and phase
2 SAs. Implementations should be able to handle this condition,
although some traffic loss may be unavoidable under this condition.)

Since the expiration times of the phase 1 SA at each end may not be
the same, any device that gets a phase 1 negotiation should abandon
the phase 1 SA that it already has with the peer, once the new SA
has been authenticated. The authenticated ID information is
necessary to determine if the new phase 1 SA is identical to an
existing phase 1 SA.

The existence of the Initial Contact notification determines whether
it should delete any phase 2 SAs it has with the peer.

Therefore, the rules for phase 1 re-keying are:

  Initial Phase 1 Negotiation:
   -responder deletes any pre-existing phase 1 SA with the peer when
  authentication of peer complete
   -initiator uses Initial Contact notification
   -responder may also use Initial Contact notification
   -responder deletes all phase 2 SAs with the peer

  Phase 1 Expiration:
   -Delete notification may be sent only if permanent deletion of
  the phase 1 SA (and all its phase 2 SAs) is intended

  New Phase 1 Negotiation:
   -responder deletes any pre-existing phase 1 SA with the peer when
  authentication of peer complete
   -no Initial Contact notification; phase 2 SAs are kept
   -if attempt fails, all other SAs are also deleted (no Delete
  notification is used, since there is no valid SA)
   -initiator should sent delete notification

  Maximum of one Phase 1 SA between peers (except during SA set-up)

 Note that any information that may be associated with pre-existing
 phase 1 SAs should be carried over into the new SA. Examples of this
 type of information are server addresses passed during using the
 Configuration Exchange mode.

3.1.1 Initial Contact Notification

  As stated above, the initial contact notification should be used
  only on the very first phase 1 that is negotiated between two peers.

  If used on subsequent negotiations, it means that all pre-existing
  SAs (phase 1 and phase 2) held between the peers should be deleted.

  This is the mechanism used to detect when an SA end point has
  crashed and is now alive again, for example.

3.1.2 Delete Notification

  As currently defined by the IPSec documents, this notification is an
  advisory only and is optional and unacknowledged.

  Given that it is optional, UDP based, and not used by some existing
  implementations, it should never be considered necessary. Further,

its value is debatable, especially given that explicit SA re-keying
rules are being used.

Further, reception of a Delete notification for phase 1 should not
be used before re-keying, since the phase 1 SA is being re-keyed,
not deleted. It should be used only to indicate permanent deletion
of a phase 1 SA and all phase 2 SAs created by it.

Even though its use is of dubious value, it should be sent when
permanent deletion of phase 1 SAs is intended, if only as a place-
holder for the proposed Delete mode for IPSecond.

3.1.3 Re-keying Timing

To reduce the probability of simultaneous re-keying, each device
should re-key at a variable time with respect to the SA's expiration
time, in case they are the same. These recommendations apply to both
phase 1 and phase 2 SAs.

Examples of this include:

1) Re-keying at a random percentage of the lifetime of the SA,
    such as 75% to 90%.

2) The end with the higher SPI re-keying at 95% of the lifetime,
    while the end with lower SPI re-keying at 85% of the lifetime.

In any case, simultaneous attempts at re-keying should be supported
in one form or another, since it can never be guaranteed that this
will not happen.

4. IPSecond Recommendations

The recommendations made in sections 2 and 3 of this document have
limitations in their ability to provide lossless, reliable and
interoperable SA re-keying due to restrictions of existing
implementations and the existing IPSec documentation.

This section makes recommendations for explicit re-transmission
rules, phase 1 and phase 2 re-keying and introduces a new mode for
reliable SA deletion in order to better provide reliable, lossless
and interoperable re-keying.

4.1 **Re-transmission Rules**

   In systems that use an even number of exchanges, the rules for re-
   transmission are relatively obvious. Simply put, a packet is re-sent
   if the expected response to it is not received within a certain
   period of time.

   However, IPSec has a number of modes that have an odd number of
   packets. This can lead to confusion as to when the re-transmission
   rules should be applied. This in turn can lead to the dropping of
   aggressive and quick modes' third messages. It is recommended that
   each of these modes have specific rules applied to them to avoid
   issues these issues.

   These rules will be applied based on request-response pairs. Packets
   are defined as a request or a response in an exchange. The requestor
   is responsible for re-sending the request in order to solicit the
   response. The responder (not to be confused with an SA negotiation
   responder) is responsible for re-sending the response as it receives
   the initial and subsequent transmissions of the request.

   In each of the cases of modes with an odd number of packets, the
   request-response pair must be applied across the odd number of
   packets. This means that at least one packet must be considered the
   response to the previous packet, and must also be considered the
   request of the next request-response pair.

   This means that an implementation must be able to perform re-
   transmission of packets after it normally would have considered
   itself to be done with an exchange or a mode. Further, any timers
   set by the transmission of the final message of an exchange should
   be reset when re-transmission occurs.

4.1.1 **Aggressive Mode Re-Transmission Rules**

   In aggressive mode, the second message is the message that is both a
   response and a request. Therefore, the responder in a phase 1
   negotiation that uses aggressive mode must re-transmit the second
   aggressive mode message to solicit a third aggressive mode message
   that it perceives as lost.

4.1.2 **Quick Mode Re-Transmission Rules**

   In quick mode, the second message is the message that is both a
   response and a request. Therefore, the responder in a phase 1

negotiation must re-transmit the second quick mode message to
solicit a third quick mode message that it perceives as lost.

[4.2](#) SA Delete Mode

The purpose of the SA Delete mode is to unambiguously delete SAs
used as pairs. It is called a mode for syntactical consistency with
quick mode, new group mode and so on.

The Delete request notification's format is the same as the Delete
notification, and may or may not refer to multiple SAs. It is
interpreted to mean the following:

  "I am not sending anymore traffic on this SA pair (or these SA
  pairs). Would you please stop sending traffic on it (or them), and
  send me an Delete acknowledgement when you are done?"

The receiver of the Delete request then switches his outbound
traffic to another SA (the next oldest), deletes both inbound and
outbound SAs and sends the Delete acknowledgement.

This is interpreted to mean:

  "I am also not sending anymore traffic on this SA pair (or these
  SA pairs). You may delete it (or them)."

The receiver of the Delete acknowledgement may then delete the
inbound SA. The outbound SA should have already been deleted or
somehow not used before the sending of the Delete request.

Note that re-transmission rules apply to the request-acknowledge
pair. That is, if the initiator of the Delete mode does not get the
Delete acknowledgement, the Delete request should be re-transmitted.
Similarly, if the responder of the Delete request receives multiple
copies, multiple copies of the Delete acknowledgement should be
sent.

If the retry counter for the Delete request expires, the SAs
indicated in the request should be unilaterally deleted.

Both messages must be sent encrypted under the protection of a phase
1 SA.

Note that there is a race condition for the Delete request and
Delete acknowledgement notifications if an implementation sends them
immediately after sending a packet on one of the SAs to be deleted.
The race occurs if the packet order gets changed in the network and

the notification packets arrive before packets sent on the SAs to
which the notifications refer.

The Delete request-acknowledgement pair should also be applied to
phase 1 SAs. In this case, the phase 1 SA is not completely torn
down until the reception of the Delete acknowledgement message.

As a specific clarification, the binding between the inbound and the
outbound SAs is not weakened. In the messages used, the SA specified
in the Delete notification is that of the sender's inbound SA. In
other words, the SPI sent to be deleted is the SPI that was
generated by the sender. This is simply to be consistent with the
format of the current Delete notification. It may be more reasonable
to specify both inbound and outbound SPIs in the SA Delete mode
messages.

Additionally, the Delete mode is used to delete phase 1 SAs as well.
In this case, the SPIs values used are the cookies of the phase 1
SA.

The introduction of this mode does not eliminate the use for the
existing Delete notification. It could still be used if an
implementation determines it needs to immediately (and impolitely)
delete an SA. Implementations must still recognise that it is sent
over UDP and may be dropped.

4.3 Phase 1 Re-keying for IPSecond

The phase 1 re-keying method described in Section 3 requires only
one change for IPSecond. That is the required use of the new Delete
mode.

The Delete mode must be used in association with phase 1 when an
implementation intends to permanently delete a phase 1 SA. This may
happen due to adminstration shut-down, policy change, remote client
session termination, re-keying failure or other reasons.

When used after a phase 1 re-keying failure, it is sent by the
initiator of the phase 1 negotiation. In this case, the Delete mode
uses the cookies of the expired phase 1 SA, rather than the cookies
of the SA negotiation that failed. It must also use the old phase 1
SA to protect the Delete mode.

The reasons for this are that the responder's phase 1 may not have
expired. The failure of the new phase 1 negotiation cannot be used
by the responder to delete its old phase 1 SA since it is likely
that authentication of the new phase 1 SA has not yet occurred.

Because of this, there is a logical overlap of phase 1 SAs that
implicitly ends upon successful negotiation of the new phase 1 SA.

4.4 Phase 2 Re-keying for IPSecond

The phase 2 re-keying proposal described in Section 2, while
necessary under the circumstances, is not the ideal method of re-
keying. It forces the specific transfer times of SAs, thus making
the intent of paragraph 2, section 9 of [IKE] impossible.

This section describes proposals related to re-keying for the next
version of the IPSec protocols. The purpose is to precisely define
re-keying so that implementations are lossless and perfectly
interoperable during re-keying. It also allows the spirit of
paragraph 2, section 9 of [IKE] to be used. Further, it meets the
requirements of paragraph 3 of section 4.3 of [ISAKMP].

A summary of the recommendations is:

1) Define and require that the normal procedure is to use the
    oldest phase 2 SA first, and to use it until its natural
    expiration.

2) Use the recommended re-transmission request rules for quick
    mode.

3) Make use of the Delete mode a requirement.

4.4.1 Oldest Phase 2 SA First

The concept of using the oldest phase 2 SA first for outbound
traffic allows the maximum use of negotiated keys and allows for the
pre-negotiation of an arbitrary number of phase 2 SAs to be made
available for later use.

The oldest SA is also defined as the first negotiated of the
available SAs.

Additionally, it decouples new phase 2 SA negotiation from old phase
2 SA deletion, and the need to transfer to the new when the old SA
is deleted.

It also eliminates the race condition that occurs during SA set up
during re-keying. This means that use of the commit bit to avoid the
race condition is not necessary except when the very first phase 2
SA is set up.

4.4.2 **Phase 2 Re-keying Illustration**

   This section illustrates the events when re-keying occurs using the
   above proposals. Note the simplifications due to the decoupling of
   SA negotiation and old SA deletion.

```
                 Initiator                  Responder

           Inbound    Outbound       Inbound    Outbound
              |           |              |           |
    1    ----------------- |             |           |
              |           |  -----------  |           |
              |           |               -------------------->  2
              |           |              |           |     |
              |           |               --------------------   3
              |           |  -----------  |           |
    4    <--------------- |              |           |
          |  |            |              |           |
    5    ----------------- |             |           |
              |  *6       |  -----------  |           |
              |  |        |               ------------------> 7
              |  |        |              |           |
              |  |        |              |   *8      |
              |  |        |              | |         |
    9
              |  |        |              | |         |
              |  |      *10 *10          | |         |
    11   ----------------- |             | |         |
              |  |            -----------  | |         |
              |  |                         -------------------> 12
              |  |                        |           |
              |  |                        | |  *13 * 13
              |  |                   14 * |           |
              |  |                        |           |
              |  |                         --------------------  15
              |  |            -----------  |           |
    16   <--------------- |             |           |
              |  |            |              |           |
           *17|            |              |           |
              |            |              |           |
```

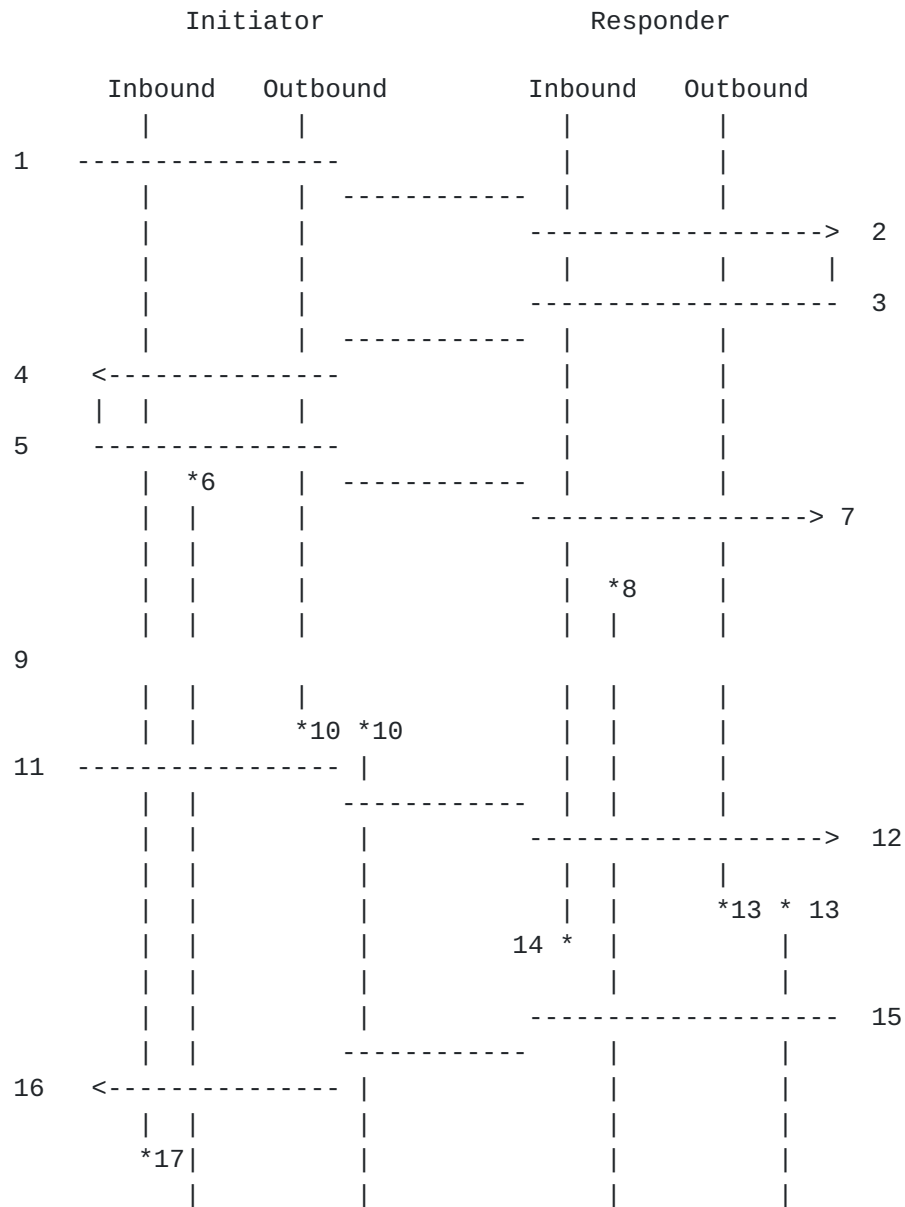   Figure 4-1 Recommended IPSecond Phase 2 Re-key Sequence Chart,
                          Initiator Expiration

   1) Initiator sends first quick mode message.

   2) Responder receives first quick mode message.

3) Responder sends second quick mode message.

4) Initiator receives second quick mode message.

5) Initiator sends third quick mode message.

6) Initiator sets up new inbound SA. Implementations may choose to
    set up the new outbound SA at this time, as long as they do not
    use it.

7) Responder receives third quick mode message.

8) Responder set up new inbound SA. Implementations may choose to
    set up the new outbound SA at this time, as long as they do not
    use it.

9) Initiator's old SA pair expires.

10) Initiator starts using new outbound SA and stops using old
    outbound SA.

11) Initiator sends first SA Delete mode message.

12) Responder receives first SA Delete mode message.

13) Responder sets up new outbound SA.

13) Responder deletes old outbound SA and starts using new outbound
    SA.

14) Responder deletes old inbound SA.

15) Responder sends second SA Delete mode message.

16) Initiator receives second SA Delete mode message.

17) Initiator deletes old inbound SA.

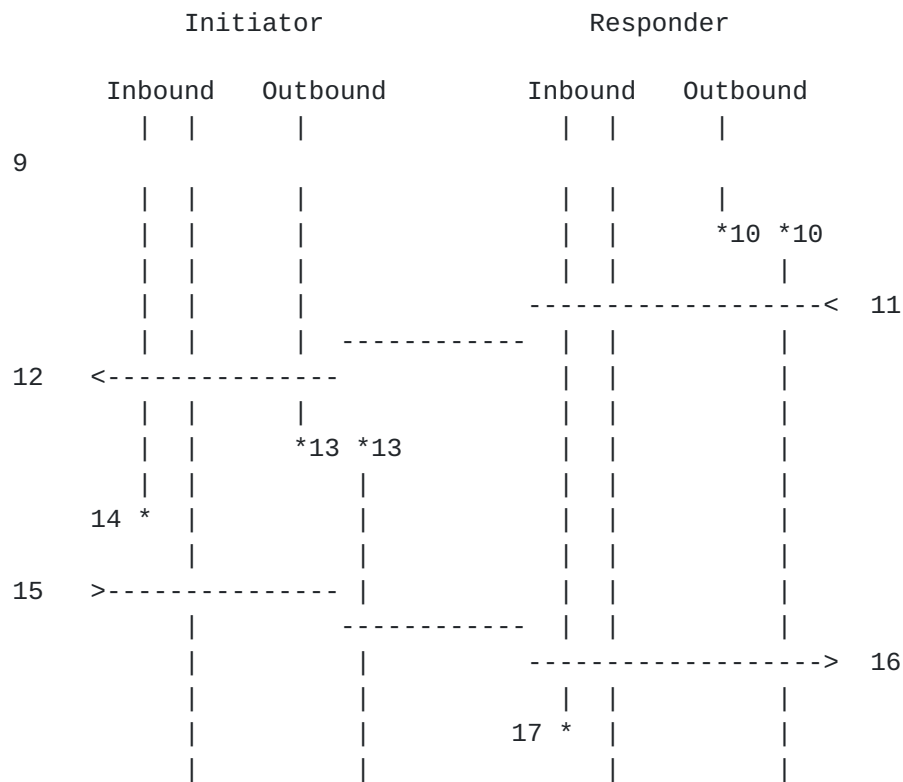If the responder's old SA expires first, the events are as follows.

```
                 Initiator                  Responder

           Inbound    Outbound        Inbound    Outbound
             |  |         |              |  |         |
     9
             |  |         |              |  |         |
             |  |         |              |  |       *10 *10
             |  |         |              |  |         |
             |  |         |            ------------------<   11
             |  |         |  -----------   |  |         |
     12    <---------------              |  |         |
             |  |         |              |  |         |
             |  |       *13 *13          |  |         |
             |  |         |              |  |         |
         14 *  |          |              |  |         |
             |            |              |  |         |
     15    >--------------- |            |  |         |
             |          ------------   |  |         |
             |            |          ------------------->   16
             |            |            |  |         |
             |            |         17 *  |         |
             |            |              |  |         |
```

                Figure 4-2 Recommended IPSecond Phase 2 Re-key Sequence Chart,
                                Responder Expiration

9) Responder's old SA pair expires.

10) Responder starts using new outbound SA and stops using old
    outbound SA.

11) Responder sends first SA Delete mode message.

12) Initiator receives first SA Delete mode message.

13) Initiator sets up new outbound SA.

13) Initiator deletes old outbound SA and starts using new outbound
    SA.

14) Initiator deletes old inbound SA.

15) Initiator sends second SA Delete mode message.

16) Responder receives second SA Delete mode message.

17) Responder deletes old inbound SA.

[5](#). Acknowledgements

   Some of the concepts presented in this memo are based on work done
   by TimeStep Corporation's engineering group.

   Others are taken from concepts discussed within the IPSec working
   group.

[6](#). References

   [IKE] Harkins, D., Carrel, D., "The Internet Key Exchange (IKE),"
          [draft-ietf-ipsec-isakmp-oakley-08.txt](#).

   [ISAKMP] Maughan, D., Schertler, M., Schneider, M., and Turner, J.,
          "Internet Security Association and Key Management Protocol
          (ISAKMP)," [draft-ietf-ipsec-isakmp-10](#).{ps,txt}.

Security Considerations

   This document is associated with the IPSec family of documents. As
   such, security considerations permeate the document.

Author's Address

   Tim Jenkins
   tjenkins@timestep.com
   TimeStep Corporation
   362 Terry Fox Drive
   Kanata, ON
   Canada
   K2K 2P5
   +1 (613) 599-3610

   The IPSec working group can be contacted via the IPSec working
   group's mailing list (ipsec@tis.com) or through its chairs:

Robert Moskowitz
rgm@icsa.net
International Computer Security Association

Theodore Y. Ts'o
tytso@MIT.EDU
Massachusetts Institute of Technology