                      **IPsec Re-keying Issues**
                   **<draft-jenkins-ipsec-rekeying-05.txt>**


   This document is an Internet-Draft and is in full conformance with
   all provisions of Section 10 of RFC2026.


Status of this Memo


   Informational

   This memo provides information for the Internet community. This memo
   does not specify an Internet standard of any kind, nor is it
   intended to specify an Internet standard. Future considerations
   related to Internet standards are the opinions of the author, and
   not the IPsec working group.

   Distribution of this memo is unlimited.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups.  Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   Internet-Drafts are draft documents valid for a maximum of six
   months and may be updated, replaced, or made obsolete 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."

   The list of current Internet-Drafts can be accessed at
   <http://www.ietf.org/ietf/1id-abstracts.txt>.

   The list of Internet-Draft Shadow Directories can be accessed at
   <http://www.ietf.org/shadow.html>.

Copyright Notice

Table of Contents

## 1. Introduction

This document discusses issues associated with the use of protocols developed in the IETF's IPsec working group, specifically, RFC 2409 [IKE] and RFC 2408 [ISAKMP]. It is expected that the reader is familiar with those documents.

As stated earlier, this document does not specify standards of any kind, and is intended as information for IPsec implementers.

This document has three primary objectives.

The first objective is to illustrate problems and issues associated with re-keying within the confines of the current set of IPsec documents. For a number of reasons, re-keying in IPsec has become problematic, such that IPsec implementations can drop packets 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 second objective of this paper is to propose methods of performing phase 2 re-keying for IPsec implementations in such a way as to minimise packet loss and to maximise compatibility. Again, the primary focus for this is in virtual private networking (VPN) and remote access (RA) applications.

The initial focus of the first two objectives is 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 the initial version of the IPsec documents.

Finally, the third objective of the document is to provide recommendations for the next version of the IPsec protocols. These recommendations are made to best solve the re-keying problems in a manner that is not possible within the constraints of the existing IPsec documents.

The document also discusses other issues related to SA negotiation, such as SA deletion, packet acknowledgement and the commit bit.

## 2. Phase 2 SA Re-keying

This section discusses phase 2 re-keying issues and makes recommendations to minimise the impact of these issues within the current IPsec document set.

It is assumed that the purpose of re-keying is that the
implementation wants to allow the transfer of traffic processed by
the phase 2 SAs from the current phase 2 SA to a replacement phase 2
SA in such a way as to minimise the loss of user traffic.

It does not preclude the use of idle-timeouts, heartbeats or keep-
alives, resource management or other SA management techniques, as
may be required by the specific application of IPsec. Nor does it
make any specific recommendations about if or when implementations
should initiate re-keying.

It also does not assume that the implementation has specific
knowledge about the peer's behaviour. In other words, the peer's
behaviour is assumed to be any of those that may be potentially
allowed by the documents.

## 2.1 Phase 2 Re-keying Issues

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

1)  There is no specification explicitly defining how the transfer
    of traffic from old to new SAs is to be done.

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

3)  Some 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.

RFC 2401 provides only this guidance (from section 4.4.3):

NOTE: The details of how to handle the refreshing of keys
when SAs expire is a local matter.  However, one
reasonable approach is:

...

(b) There SHOULD be two kinds of lifetime -- a soft
lifetime which warns the implementation to initiate
action such as setting up a replacement SA and a
hard lifetime when the current SA ends.

As such, it makes no recommendations as to how traffic should be
moved to the replacement SA.

## 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 within a
phase 1 SA, and 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 traffic in preference to the old SA. Extending this logic may have caused implementations to abandon the old SA as soon as possible.

This interpretation of [ISAKMP] is in direct conflict with the usage implied by [IKE], resulting in potential problems.

### 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. The behaviours listed below are not necessarily mutually exclusive.

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 quick mode packets are not lost, it can be seen that interoperability is not always possible with some 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 quick mode message.

```
            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
```
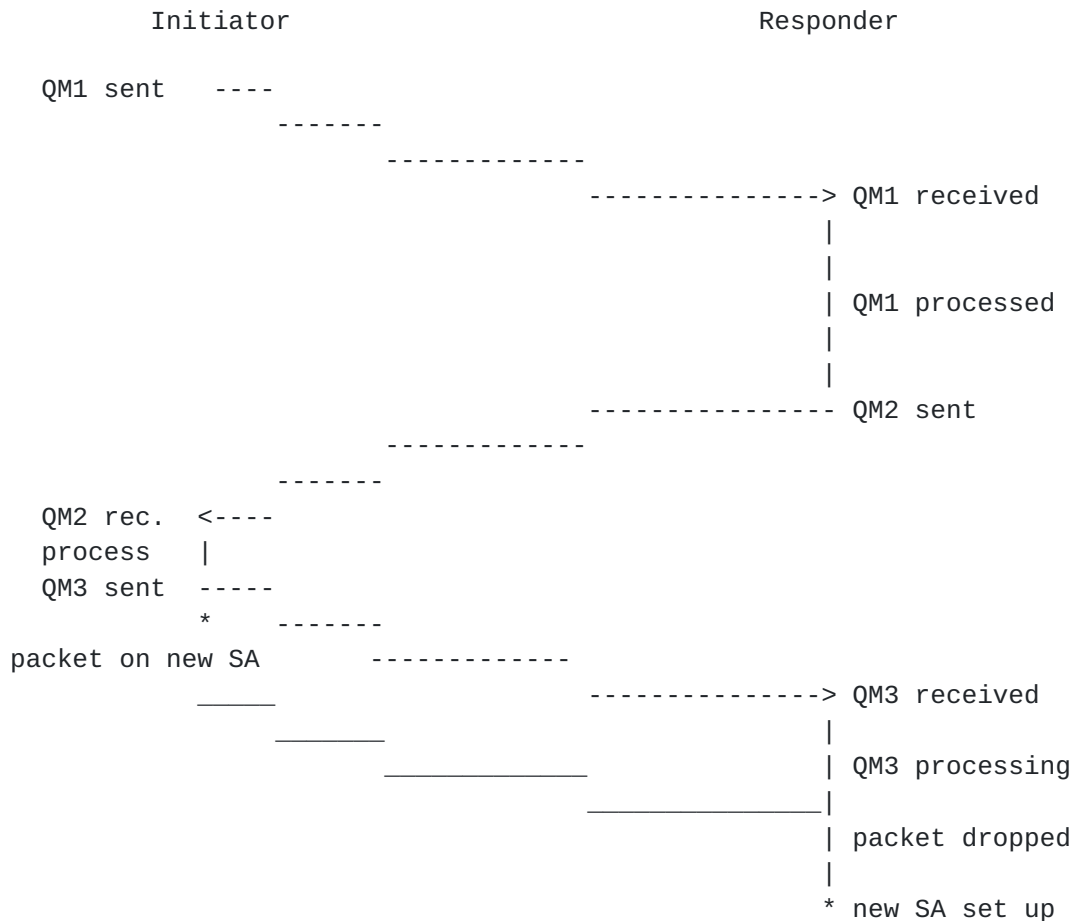
            Figure 2-1 Race Condition Sequence Chart

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 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. Nor can it eliminate race conditions due to path differences or processing times.

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 behaviours 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 ([ISAKMP]) specifies its use 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.

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 a number of possible denial of service attacks on IKE, this is not an excuse to leave the existing implementation as it is.

4)  There is no defined way to recover from the loss of the CONNECTED notification.

5)  Some implementations are using the commit bit for the wrong reasons.

The working group is addressing point 1; future versions of the IPsec documents should clarify these issues. [IKEbis] has gone a long way in clarifying this issue.

Point 3 happens because the commit bit is in the ISAKMP header, and the ISAKMP header is not authenticated, so the commit bit is susceptible to undetectable modification.

Point 5 above needs some elaboration. In a previous section, it was mentioned that the loss of the third quick mode message could cause problems, since the responder will not set up the SA at all. Because

of this, some implementations have chosen to set the commit bit as a mechanism to force the re-transmission of the third quick mode message.

This is wrong for two reasons. First, it is not the stated purpose of the commit bit. The purpose of the commit bit is to delay the usage of an SA, for whatever reason. This implies that it is not a good mechanism to cause re-transmission of the third quick mode message.

Secondly, it does not solve the packet loss problem; it only defers it. The logic of the improper usage is that the initiator will re-send the third quick mode message until it receives the CONNECTED notification (which is now effectively the fourth quick mode message).

The problem with this is that it leaves no mechanism for demanding the re-transmission of the CONNECTED notification itself. It can be dropped just as the third quick mode message can. This means that the problem that was intended to be solved by the use of the commit bit is simply pushed out to being the problem of solving the dropped CONNECTED notification.

Sections 2.2.2.1 and 4.1 describe a mechanism for solving the dropped third quick mode message problem.


## 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. By using this information, it allows the responder to set up the new inbound SA before having received the third quick mode message.

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

Additionally, it does not require use of the CONNECTED notification for prevention of the race condition, or the use of the DELETE notification for removal of the old SA. This is important since, even if they are always sent, they are unacknowledged UDP packets and may be lost.

### 2.2.1.1 Normal Conditions

Figure 2-2 shows the operation under normal (successful) conditions.

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.

```
                Initiator                 Responder

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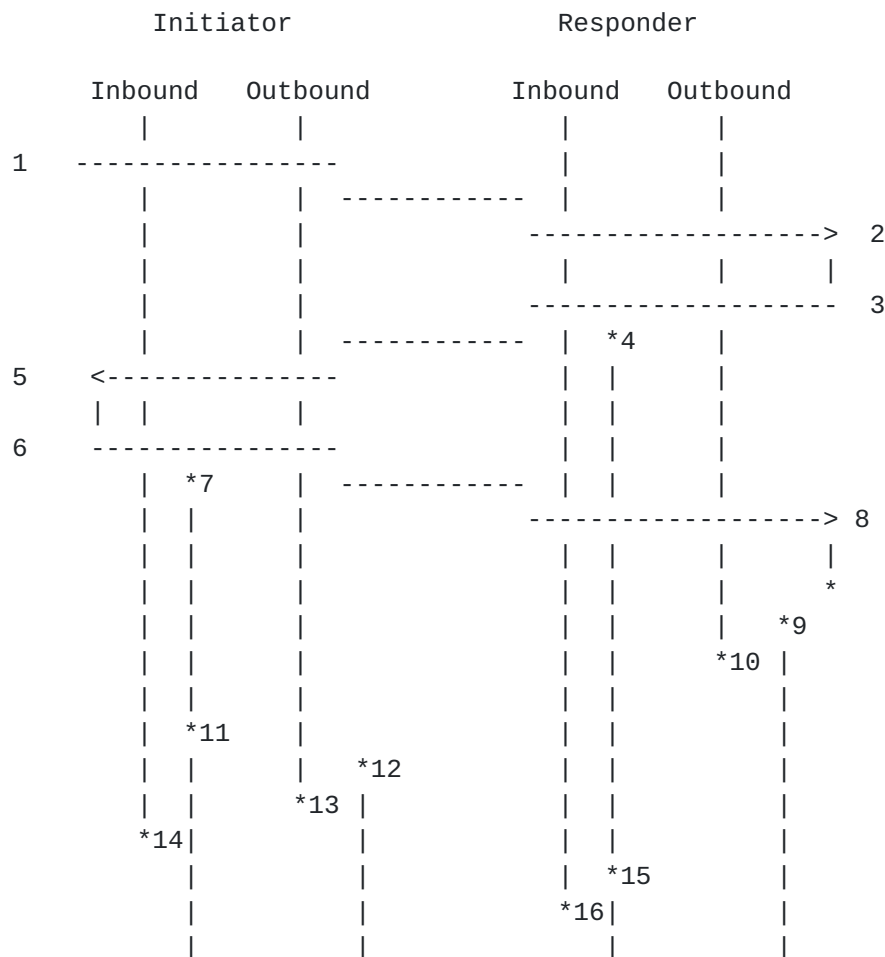
                Figure 2-2 Phase 2 SA Pre-Setup Sequence Chart

   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.


**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 2-2.

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 2-2.

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.

Note that this security hole is exactly what would result from an
attacker replaying the first quick mode message of an exchange.

### 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.

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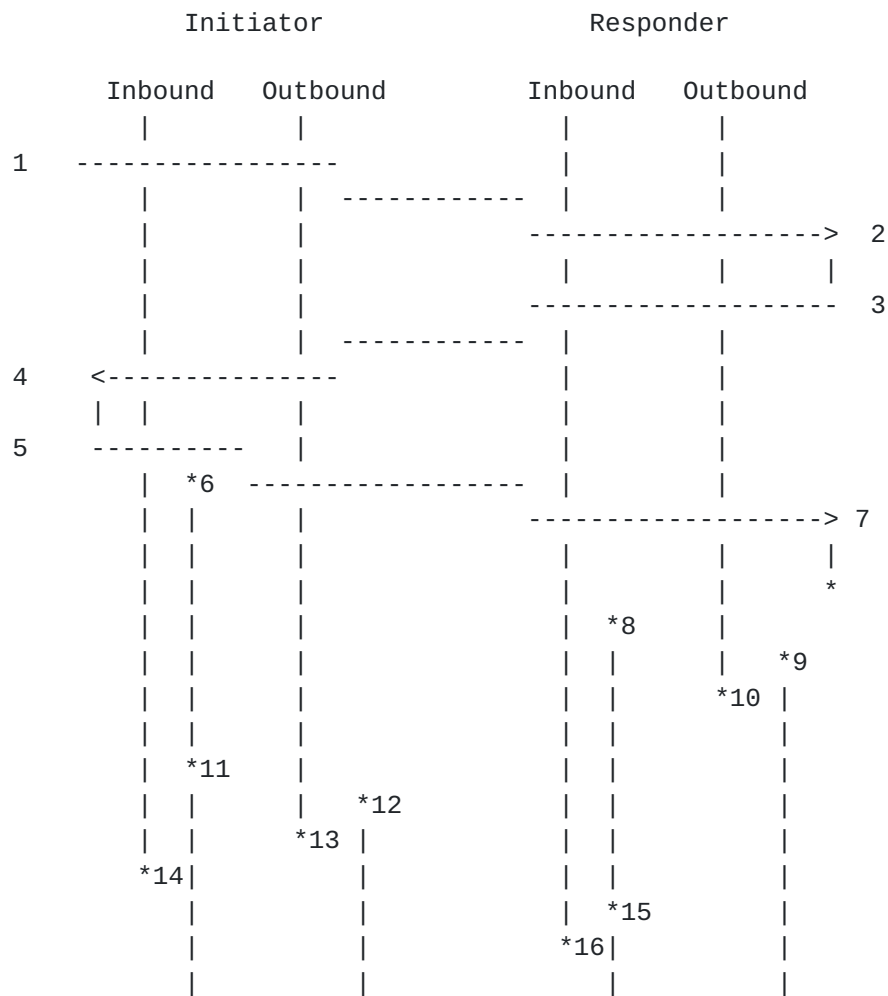SA on different and slower network paths is desired.

```
               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 sets 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.


### 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 designed to use the old SA until its
    natural expiration.

This implies that implementations must be able to respond to the re-
transmission of the second quick mode message even after having sent
the third quick mode message.


### 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 in the absence of traffic. 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 responders use the proposed method and the initiator is an implementation that uses the new SA immediately, there is no difference in SA transfer performance compared to the responder also using the new SA immediately. This is because the proposed method tries to use the new SA immediately on inbound, so it will be ready to receive on the new SA just as fast as an implementation that starts using the new immediately under all conditions. However, since the initiator is also using the new SA immediately, there is a possibility that packets will arrive at the responder on the new SA before the responder has time to set up the new SA.

When the initiator uses the proposed method, the performance (packet loss when transferring to the new SA) 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

If the responder sets the commit bit with this proposal, some of the problems described in Section 2.1.4 may occur. To reduce the effects of these problems, following rules should be followed:

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)  Sensing of traffic 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 its new inbound SA. However, this
works only if there is traffic, so it cannot be considered a useful
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.


**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 ageing 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 for
interoperability within the framework of the current IPsec documents
without compromising security.


**3**. **Phase 1 SA Re-keying**

This section discusses phase 1 SA re-keying. This proposal is
necessary for many of the same reasons a phase 2 SA re-keying
proposal is necessary.

1)  As mentioned above, the rules for phase 1 SA re-keying are not
    specified in the drafts.

2)  Adhoc implementations have lead to possible interoperability
    issues.

This section discusses potential requirements of phase 1 re-keying,
and presents some options and issues associated with those options.


## 3.1 Phase 1 SA Re-keying Requirements

Two reasons for re-keying a phase 1 SA are for freshness (time or
other) of the phase 1 SA keying material (affecting its ability to
protect phase 2 SA negotiations and to generate phase keying
material) and for re-authentication (and therefore authorisation) of
the encrypting devices.

The authorisation lifetime restriction reason stated above was
inferred as necessary due to the following paragraph from
section 4.4.3 of RFC 2401 (the Security Associations being discussed
are phase 2 SAs):

  o Lifetime of this Security Association: a time interval after
    which an SA must be replaced with a new SA (and new SPI) or
    terminated, plus an indication of which of these actions
    should occur.  This may be expressed as a time or byte count,
    or a simultaneous use of both, the first lifetime to expire
    taking precedence. A compliant implementation MUST support
    both types of lifetimes, and must support a simultaneous use
    of both.  If time is employed, and if IKE employs X.509
    certificates for SA establishment, the SA lifetime must be
    constrained by the validity intervals of the certificates,
    and the NextIssueDate of the CRLs used in the IKE exchange
    for the SA.  Both initiator and responder are responsible for
    constraining SA lifetime in this fashion.
    [REQUIRED for all implementations]

Note particularly the lifetime constraint comments in the last two
sentences.

However, this restriction reason stated above has been deemed
unimportant by the working group as a factor in determining how
phase 1 SAs are used and re-keyed for two reasons:

1) System administrators understand IPsec well enough to configure
   the combination of phase 1 and phase 2 SA lifetimes such that

terminating phase 2 SAs when authentication ends means the
unauthorised usage period is insignificant.

2) Many implementations will be required to produce a mechanism to
   tear down SAs created by entities that are no longer authorised.
   This is considered manual intervention, and thus not important
   for normal unattended operation.

The net result of this is that phase 1 SAs do not need to be
overlapped to provide a continuous indication of peer authorisation
to allow phase 2 SAs to continue to exist.

Therefore, the only reason to re-key phase 1 SAs is due to keying
material expiration. Further, it means that phase 1 SA and phase 2
SA lifetimes are unbound; that is, there are no requirements that a
phase 1 SA exist between two peers that have phase 2 SAs.

However, some applications may consider it advantageous to attempt
to keep a valid phase 1 SA up between peers at all times. This would
require overlapping of phase 1 SAs, and re-keying of old SAs before
they expire. However, these implementations must be aware that peers
may not be trying to do this, and in fact may be trying to reduce
unused resource requirements by deleting the phase 1 SA.

Factors to consider in determining how to re-key phase 1 SAs that
are not RFC-based requirements are resource issues (memory
requirements versus complex calculation requirements), SA usage with
respect to time (normal SA usage very short lived, for example),
reliability requirements, and other possible application specific
factors.


## [3.2](#) Phase 1 Re-keying Operation

Summarised, the procedure for phase 1 re-keying is:

  Initial Phase 1 SA Negotiation:
   -initiator must use INITIAL-CONTACT notification
   -responder should use INITIAL-CONTACT notification (when
  possible)
   -responder deletes any pre-existing phase 1 SA with the peer when
  authentication of peer is complete (in cases of simultaneous
  initiation, the other "new" phase 1 SA should not be deleted)
   -responder deletes all previously existing phase 2 SAs with the
  peer, if any

  Phase 1 SA Expiration:
   -DELETE notification should be sent for phase 1 SA only

      -phase 2 SAs between peers are left untouched

    New Phase 1 SA Negotiation:
      -initiator must not use INITIAL-CONTACT notification
      -responder must detect that this is a re-key and must not use
    INITIAL-CONTACT notification
      -no INITIAL-CONTACT notification is used by either end, so phase
    2 SAs are kept

    Phase 2 SA Ages, and no existing phase 1 SA
      -attempt New Phase 1 SA Negotiation
      -if that succeeds, attempt new phase 2 SA negotiation

   If an implementation wants to overlap phase 1 SAs, the following
   procedure is recommended:

    Phase 1 SA Ages:
      -peer that first detects this or desires overlapping phase 1 SAs
    negotiates new phase 1 SA; becomes new initiator
      -responder should mark any existing phase 1 SAs as re-keyed, so
    as to not re-key again if it also desires overlapping phase 1 SAs


**[3.3](#) A Note About Overlapping Phase 1 SAs**

   An earlier version of this document promoted having overlapping
   phase 1 SAs at all times. This was presented as the continuous
   channel model. Continuous channel implementations are those
   implementations that attempt to always maintain at least one valid
   phase 1 SA between any peers that have phase 2 SAs.

   The reasons and advantages of this method are discussed here.
   However, it must be re-iterated that there are no RFC-based
   requirements that implementations follow this model. In fact,
   implementations should not insist on this model due to the
   possibility that the peer may be attempting to minimise resource
   usage.

   The continuous channel method is implicitly recommended by RFC 2408.
   The following quote is from paragraph six of [ISAKMP]:

     Third, having an ISAKMP SA in place considerably reduces the cost
     of ISAKMP management activity - without the "trusted path" that
     an ISAKMP SA gives you, the entities (e.g.  ISAKMP servers) would
     have to go through a complete re-authentication for each error
     notification or deletion of an SA.

The primary advantage of the continuous existence of the logical channel is that it allows cleaner management of phase 2 SAs, particular if the two entities become unsynchronised for any reason.

It is useful whenever premature termination of communications occurs when the phase 1 SAs cannot be re-created. These conditions occur potentially under the following conditions:

1) A time-based policy is used that restricts user access to specific hours of the day, and only phase 1 authorization catches this.

2) A user's permissions are totally revoked.

3) A token card-based user removes the token card from the system.

4) Some other policy or configuration change.

A specific application for this model that provides distinct advantages is with the use of token cards. For example, if a userÆs phase 1 authentication and authorisation is bound by the presence of a token card in a reader, the removal of the card should result in all SAs being torn down. Since there exists a continuous channel, delete notifications (acknowledged or not) can be sent for all SAs when the token card is removed from the system. However, if the phase 1 SA was allowed to be deleted without being re-keyed, the local end can only unilaterally delete its SAs, leaving the two end points out of sync with each other. (It cannot send delete notifications since the absence of the card makes it unable to re-establish a phase 1 SA.)

Depending on the application, the above cases may fall under the category of special events, and thus not having significant weight when determining if the continuous channel model is the correct implementation to be used.

The continuous channel model also allows a responder to initiate re-keying under conditions where its SAs expire before the initiator's and configuration does not allow it to normally initiate to the peer. This situation is currently permitted by the RFCs if implementation should choose to not send or support the use of the RESPONDER-LIFETIME notification when the initiator's proposal has longer lifetimes than the responder is willing to accept.

Also, this model more closely ties endpoint authorization to phase 2 SA lifetime.

The disadvantages of the continuous channel model of implementation are that it uses more resources (always keeps a phase 1 SA, and potentially uses more key generation), and is slightly more complex to implement.

Finally, it must be aware of implementations that do not want or need phase 1 SAs that are continuously available.

### 3.3.1 Identity Perfect Forward Secrecy

Allowing the use of only a single phase 2 negotiation in a phase 1 SA is how identity PFS is done. This is controlled by the deletion of the phase 1 SA after a phase 2 negotiation.

In implementations that do not wish to delete all phase 1 SAs, this can be done by creating two phase 1 SAs before the first phase 2 negotiation is done. The first of these SAs is assigned the role of channel management, and thus performs SA deletion and notification transfer. The second SA is used to perform phase 2 negotiations, and is immediately deleted.

The phase 1 SA that is assigned to channel management is re-keyed to create the overlapping phase 1 SAs. Since it is the oldest phase 1 SA, it will naturally be used for all management traffic even if another phase 1 SA temporarily exists only for the purpose of performing a quick modes (see Section 3.4.1). Other phase 1 SAs are created and used to protect phase 2 negotiations and then they are immediately deleted.

Since Identity PFS is not negotiated, it may not be possible to guarantee that the peer knows Identity PFS is being used. In this case, the initiator may be required to delete its channel management SA and create a new one if the peer uses that phase 1 SA to re-key a phase 2 SA.

### 3.4 Other Phase 1 SA Re-keying Issues

This section describes other issues associated with phase 1 SA re-keying that are independent of the whether the implementation intentionally overlaps phase 1 SAs or not.

### 3.4.1 Multiple SA Usage

When there is more than one phase 1 SA between peers, it is recommended that the oldest SA be used for subsequent traffic

requiring phase 1 SAs. This allows full use of the keying material
generated and reduces race conditions. It also means that no special
expiration conditions are required when the phase 1 SAs expire by
traffic or other usage dependent expirations only, as the old SA
will eventually expire on its own due to usage.


### 3.4.2 INITIAL-CONTACT Notification

As stated above, the INITIAL-CONTACT notification should be used
only with the very first phase 1 SA 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.

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

The use of INITIAL-CONTACT may be restricted by the mode used to
negotiate phase 1 SAs. For these reasons, implementations may want
to avoid the use of aggressive mode when possible. When it is used,
it is recommended that the third aggressive mode message be
encrypted so that the INITIAL-CONTACT notification can be added to
it when needed. Note that the use of any notification by a responder
during aggressive mode is not allowed, and this document's
suggestion that the use of INITIAL-CONTACT is permitted by the
initiator if the third aggressive mode packet is encrypted is
possibly contrary to RFC2408.

Alternatively, if notifications cannot be used within the phase 1
modes at all, it is recommended that INITIAL-CONTACT be sent in a
notification packet (preferably acknowledged) immediately after the
phase 1 is complete. Reception of this notification (at any time)
should indicate to the receiver that all other SAs, phase 1 and
phase 2, with the sender must be deleted. (In other words, the SA
that was used to encrypt the notification is the only SA that is not
deleted.)


### 3.4.3 DELETE Notification

As currently defined by the IPsec documents, the DELETE notification
is 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.

However, even though its use is of dubious value, it should be sent
when any SA (phase 1 or phase 2) is deleted. Since the expiration of
SAs might not occur at the same time at both ends for a number of
reasons, use of the DELETE notification can increase the probability
that both ends are synchronised with respect to SA usage.

Further, implementations should attempt to use the acknowledged
notify exchange as described in [IKEbis].

### 3.4.4 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
limit, in case they are the same. These recommendations apply to
both phase 1 and phase 2 SAs.

An example of this is that the end with the higher IP address re-
keys at 95% of the lifetime, while the end with lower IP address re-
keys at 85% of the lifetime.

Whatever rule is chosen, it is recommended that the rule be
deterministic in order to have predictable and consistent behaviour
between peers. If the rule had used the SPI as the determining
factor (as an example did in the first version of this document),
different peers would be doing the re-keying at different times.

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 under all circumstances.

### 4. Next IPsec Version 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 describes the use of a new
mode for reliable SA deletion in order to help provide reliable,
lossless and interoperable re-keying.

Also, a replacement for the commit bit is proposed.

## [4.1](#) Re-transmission Rules

In systems that use exchanges that have an even number of packets, 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, depending how the re-transmission rules are applied to the packets in the exchange. This in turn can lead to the dropping of aggressive mode's and quick mode's third messages. It is recommended that each of these modes have specific rules applied to them to avoid re-transmission 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. Note that the responder must exist after transmitting a response in case that response is dropped.

In the 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](#) Main Mode Re-Transmission Rules

In main mode, there are effectively three completely separate exchanges. The first request-response pair contains the SA proposals, the second pair contains the keying material, and the third pair contains the authentication material. (These descriptions are generalised for the purposes of stating what the exchanges are, and are not intended to create discussion on the actual contents of the exchanges.)

As an example of the separation of the exchanges, there is no need
to re-send the second main message to solicit the third main mode
message, since the responder should not send the fourth main mode
message until receiving the third main mode message. The absence of
the fourth main mode message will cause the initiator to re-send the
third main mode message.

Keeping the exchanges separate from a re-transmission point of view
should simplify implementations.


**4.1.2** **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.


**4.1.3** **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.

These rules must apply independently of the state of the commit bit,
since there are currently no timing restrictions on the transmission
of the CONNECTED notification.


**4.2** **Acknowledged SA Deletion**

A previous version of this document described a new mode called
Delete Mode. This mode is no longer necessary, as the new proposed
Acknowledged Informational exchange can be used with the delete
payload to perform the same thing. (See section 6.4.2 of [IKEbis].)

This section (of this document) describes in detail how the
Acknowledged Informational exchange should be used when deleting
SAs.

The Acknowledged Informational exchange consists of two packets. The
first packet is the transmission of a notify or delete payload. The
second is the acknowledgement of that packet.

When used with a delete payload, it is interpreted to mean the
following:

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

The receiver of the delete request then switches his outbound
traffic to another SA, 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 (or these SAs).
  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.

Note that there is a race condition for the delete request and
delete acknowledgement packets 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 delete mode packets arrive before packets sent on the SAs to
which the deletes 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 phase 2 SAs is not weakened. In the messages used, the SA
specified in the delete request is that of the sender's inbound SA.
In other words, the SPI sent in the notification is the SPI that was
generated by the sender. When phase 1 SAs are being deleted, the SPI
values used are the cookies of the phase 1 SA to be deleted.

The use of the Acknowledged Informational 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 Future Versions of IPsec**

The phase 1 re-keying method described in Section 3 requires only
one change for future versions of IPsec. That change is the addition
of the required use of the Acknowledged Informational exchange when
deleting SAs when it cannot be guaranteed that the peer's phase 1 SA
lifetime is identical to the local lifetime.


**4.4** **Phase 2 Re-keying for Future Versions of IPsec**

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. It is also
complicated to implement.

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 Acknowledged Informational exchange a
    requirement for SA deletion.


**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.

Additionally, it decouples new phase 2 SA negotiation from old phase 2 SA deletion, and the need to transfer to the new SA during re-keying.

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.

Another advantage of being able to pre-negotiate phase 2 SAs is for applications that use large amounts of data in a period of time that would be too short for re-keying of the SA used to take place reliably.

The oldest SA is defined as the first negotiated of the available SAs. In cases of simultaneous and near simultaneous SA negotiation, the use of the acknowledged DELETE notification and the ability to overlap SAs for an arbitrary period of time should make this condition manageable.


**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, old SA deletion and the transfer of traffic from the old to the new SA.
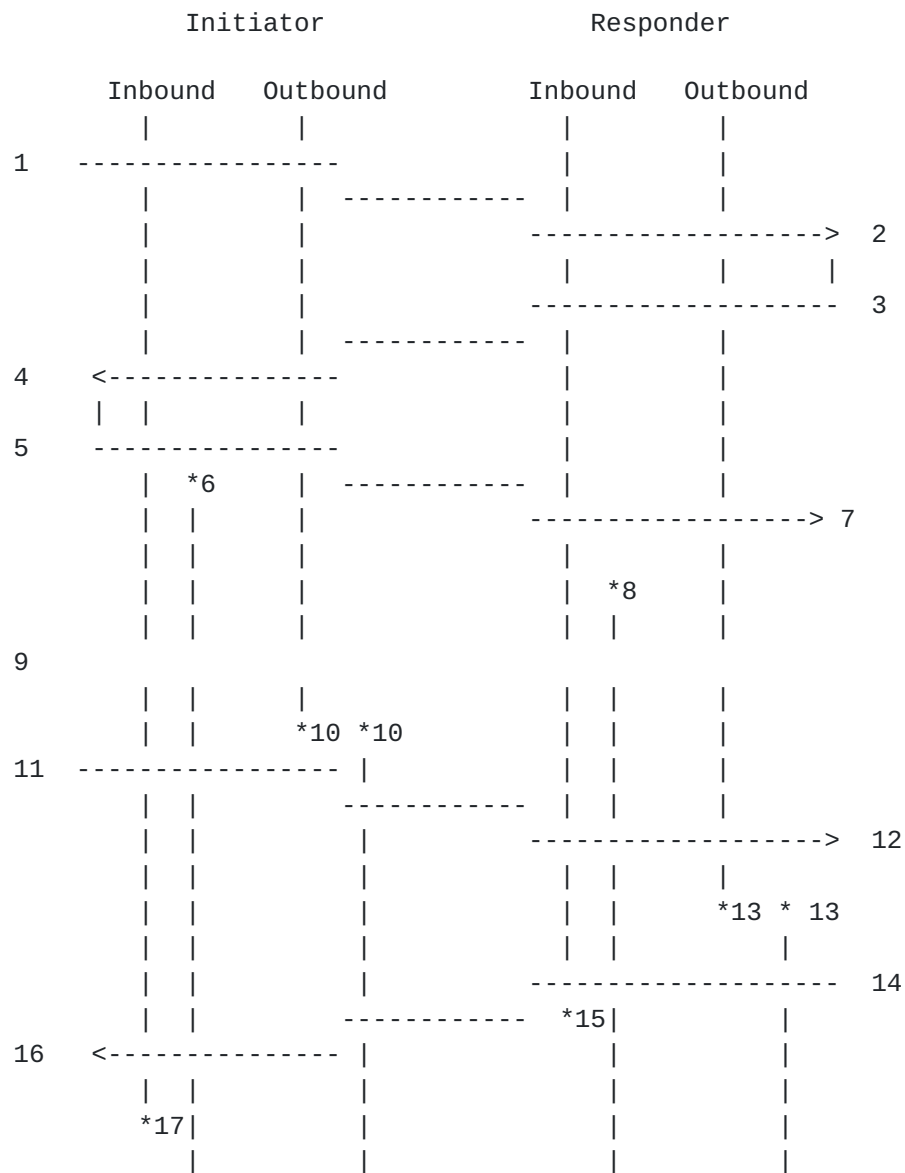
```
            Initiator                  Responder

         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 Phase 2 Re-key Sequence Chart, Initiator
                           Expiration, Future

    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 Acknowledged Informational exchange
    message with a delete payload.

12) Responder receives first Acknowledged Informational exchange
    message.

13) Responder sets up new outbound SA.

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

14) Responder sends second Acknowledged Informational exchange
    message.

15) Responder deletes old inbound SA.

16) Initiator receives second Acknowledged Informational exchange
    message.

17) Initiator deletes old inbound SA.

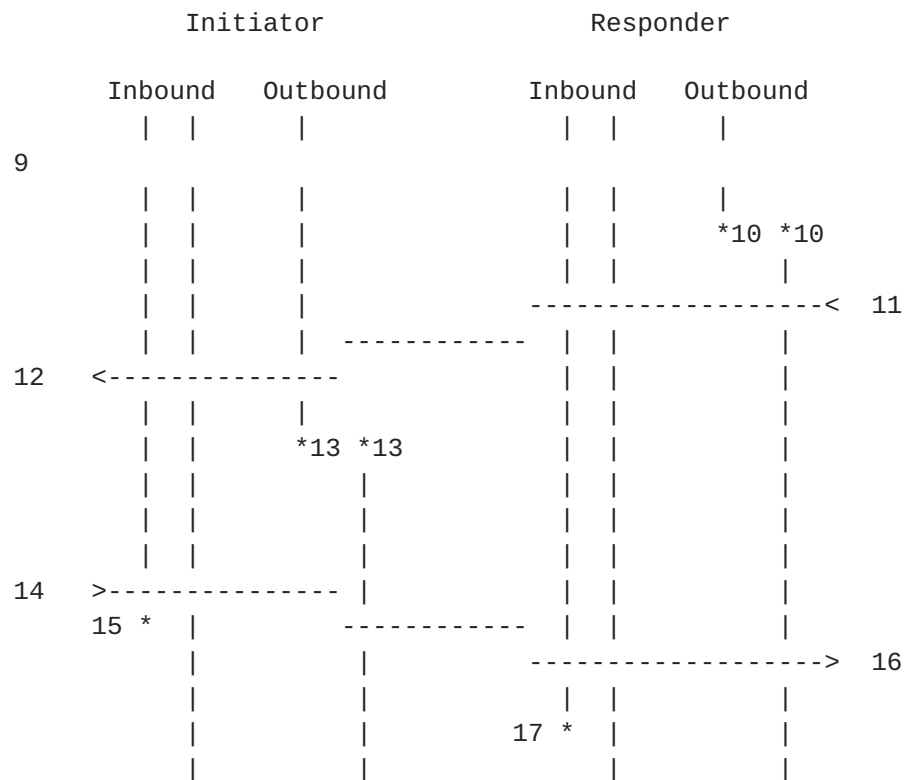If the responder's old SA expires first, the events are as follows.

```
                 Initiator                Responder

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

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

   9)  Responder's old SA pair expires.

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

   11) Responder sends first Acknowledged Informational exchange
       message with a delete payload.

   12) Initiator receives first Acknowledged Informational exchange
       message.

   13) Initiator sets up new outbound SA.

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

   14) Initiator sends second Acknowledged Informational exchange
       message.

   15) Initiator deletes old inbound SA.

16) Responder receives second Acknowledged Informational exchange
    message.

17) Responder deletes old inbound SA.


## 4.5 Commit Bit Replacement

The intent of this section is to propose a mechanism to allow
implementations to delay the usage of negotiated SAs. Its use may
eliminate the need for the commit bit, and will not suffer from any
of the problems of the commit bit. While the commit bit usage is
much better defined by [IKEbis], it is unable to solve all the
difficulties associated with it.

Replacement of the commit bit is done by the introduction of a new
mechanism to indicate to a peer that usage of a newly negotiated SA
should be deferred. Then, depending on the deferral time intended,
one of two mechanisms is introduced to indicate that the SA may be
used.

These mechanisms are preferred over the commit bit for the following
reasons:

o  They receive the full protection of phase 1 SAs, and as such
    provide the maximum resistance to denial of service attacks.

o  Their use is clearly and unambiguously defined.

o  They are resistant to the possibilities of dropped packets.


## 4.5.1 DEFER_USAGE Notify Payload

The indication that an SA should not be made available for use
immediately by a peer can be indicated by the addition of a new
notify payload to the quick mode that negotiated the SA. To allow a
single quick mode to negotiate multiple SAs, the DEFER_USAGE notify
payload explicitly names the SA whose use is to be deferred, in the
same manner as the current DELETE payload.

The DEFER_USAGE notify payload should be added by the peer wishing
to delay usage of an SA.

On reception of the DEFER_USAGE notify payload, the newly negotiated
SA should be set aside until reception of the ALLOW_USAGE notify
payload, described in the next section, or the reception of the
CONNECTED notification.

The expected response depends on which type of DEFER_USAGE
notification is sent. These types are termed long and short. A short
DEFER_USAGE notification causes a quick mode to become four messages
in length, as with the intended use of the commit bit. A long
DEFER_USAGE notification causes quick mode to proceed normally, with
usage of the specified SA deferred until the sender of the
DEFER_USAGE notification sends the ALLOW_USAGE notify.

Implementations should be prepared to receive the long DEFER_USAGE
notification for the same SA (pair) that they send it for; in other
words, usage of both SAs (inbound and outbound) of the negotiated
pairs may be deferred simultaneously by both peers.

There are no time constraints associated with the sending of the
long DEFER_USAGE notification and the subsequent reception of the
ALLOW_USAGE notification.

Usage of the short DEFER_USAGE notification is restricted to quick
mode responders only. It causes the transmission of a CONNECTED
notification as a fourth quick mode message in the same way that the
commit bit does.


**4.5.2 ALLOW_USAGE Notify Payload**

The purpose of this notify is to indicate to a peer that an SA may
now be used. Normally, usage of the SA by the peer would have been
deferred by the use of the long DEFER_USAGE notify payload,
described in the previous section. However, reception of this notify
for an SA whose usage has not been deferred is not considered an
error.

This payload MUST be used only with the Acknowledged Informational
exchange.

The initiator of the exchange must start usage of the inbound SA of
the pair when sending the first packet of the exchange. Usage of the
initiator's outbound SA must wait until reception of the
acknowledgement packet of the exchange.

The responder of the exchange must start usage of its inbound SA of
the pair before sending the acknowledgement, and may start usage of
its outbound SA of the pair any time after receiving the first
packet of the exchange.

The initiator of the exchange re-transmits the ALLOW_USAGE
notification until it receives the acknowledgement packet or exceeds
its re-try counter.

If both peers deferred use of the SA, two transactions of the
ALLOW_USAGE notification are required (one in each direction) before
the SAs involved may be used.


## 5. Acknowledgements

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

Others are taken from concepts discussed within the IPsec working
group, particularly some of the concerns expressed about problems
with the commit bit, and also expressed concerns about the
continuous channel model for phase 1 SA re-keying.

Thanks also go to those implementers who have stated the value of
this document, and made suggestions for it.


## 6. References

[IKE]   Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)",
        RFC2409, November 1998

[ISAKMP]Maughan, D., Schertler, M., Schneider, M., and Turner, J.,
        "Internet Security Association and Key Management Protocol
        (ISAKMP)", RFC2408, November 1998

[IKEbis]Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)",
        draft-ietf-ipsec-ike-01.txt, May 1999, work in progress

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@catenanet.com
     Catena Networks, Inc.
     Suite 300
     320 March Rd.
     Kanata, ON
     Canada
     K2K 2E3
     +1 (613) 599-6430


   The IPsec working group can be contacted via the IPsec working
   group's mailing list (ipsec@lists.tislabs.com) or through its chair:

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

     This document expires September 17, 2000