Internet Engineering Task Force
IP Security Working Group
Internet Draft

Tim Jenkins
TimeStep Corporation
October 20, 1999

IPSec Re-keying Issues

<draft-jenkins-ipsec-rekeying-02.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.

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

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.

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.

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 defining 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 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 in preference to the old SA. In other words, the old SA should be abandoned 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 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.

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.

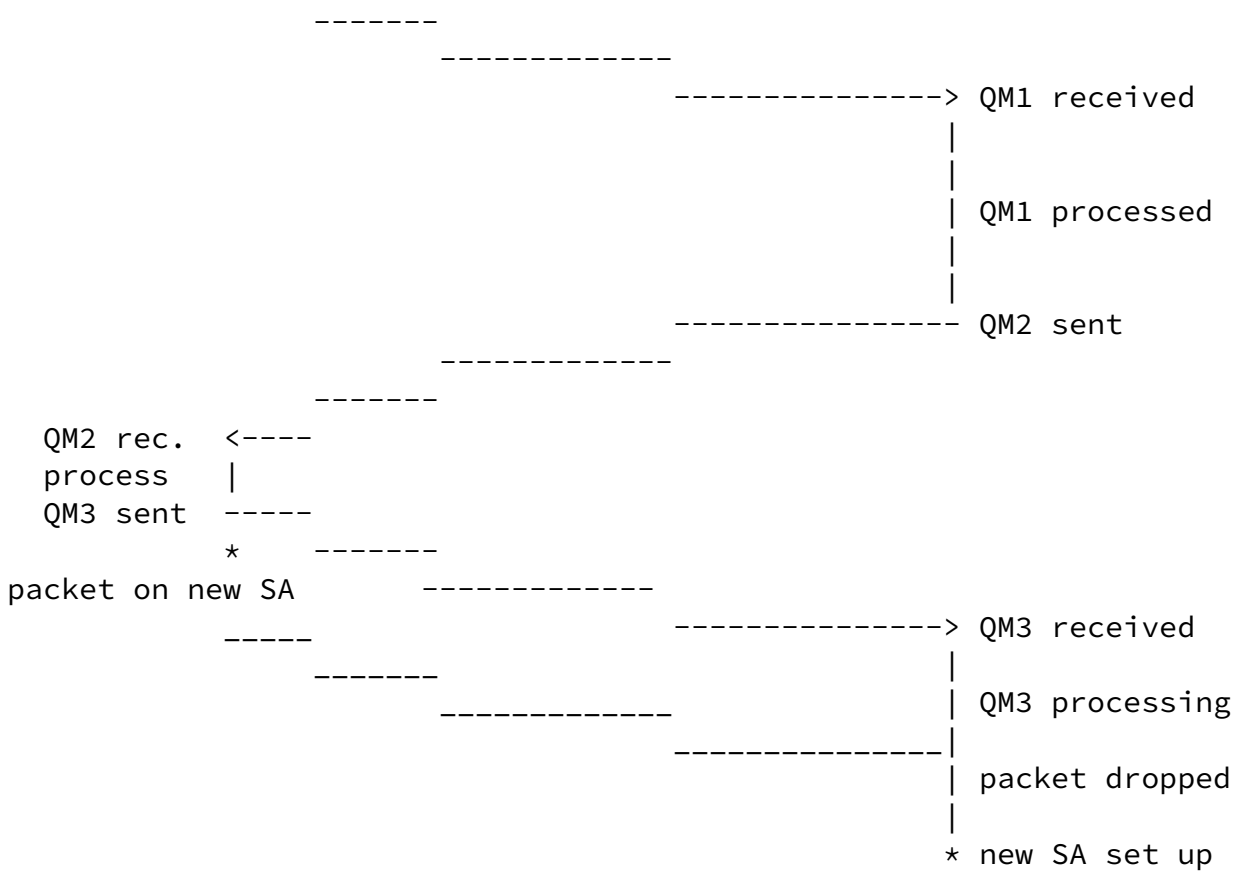


Figure 2-1 Race Condition Sequence Chart

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 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 ([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.

Point 1 is being addressed by the working group; future versions of the IPSec documents should clarify these issues. [IKEbis] has done an excellent job of clarifying this issue.

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

Point 5 above needs some elaboration. In a previous section, it was mentioned that the loss of the third quick mode message can 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 resend 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-Setup

As a starting point, the responder pre-setup 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.

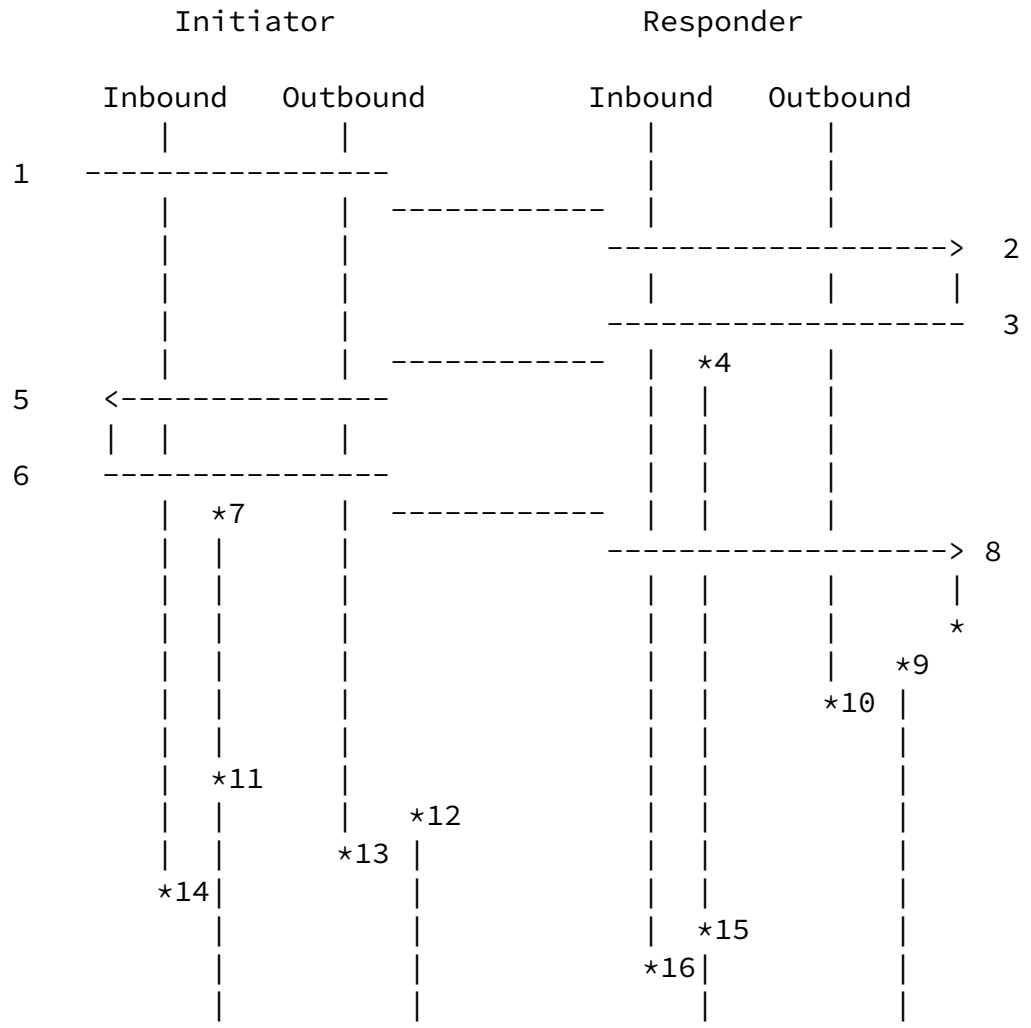


Figure 2-2 Phase 2 SA Pre-Setup Sequence Chart

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-Setup 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.

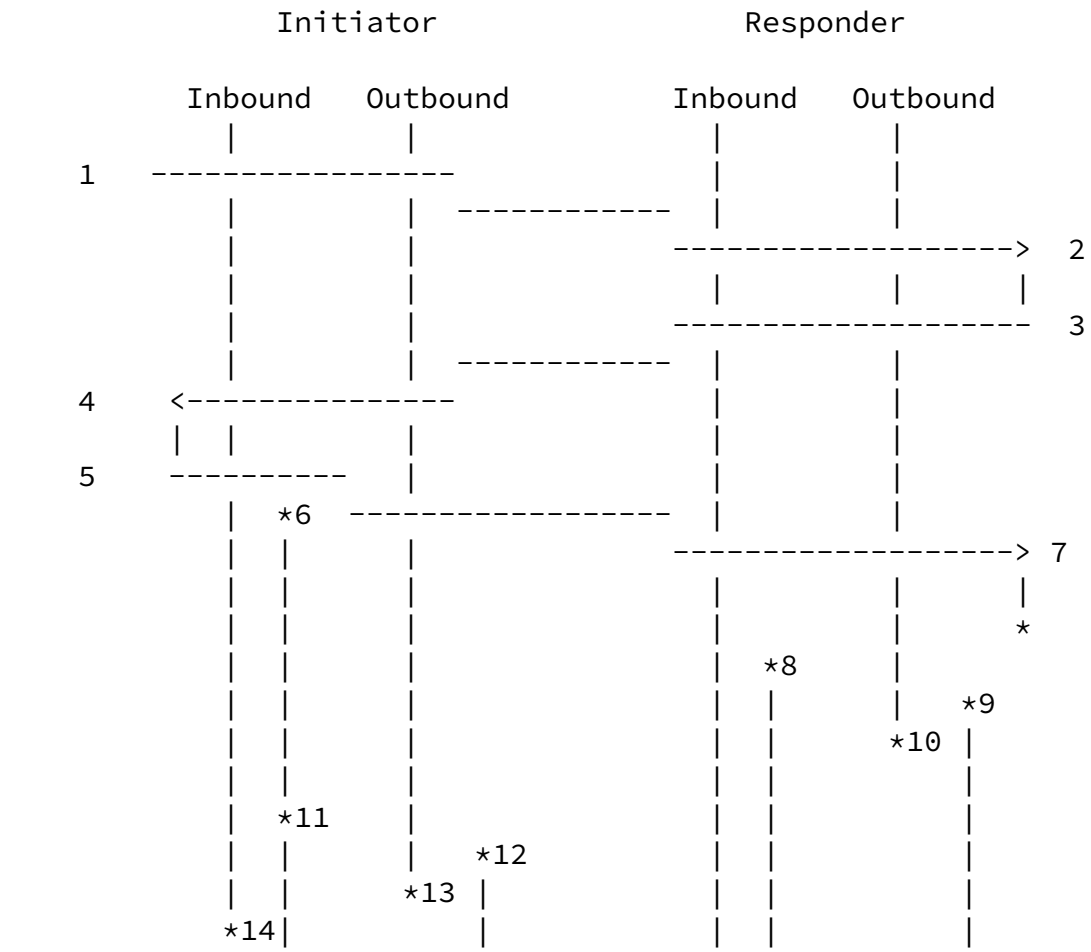

Initiator
Responder
Inbound
Outbound
Inbound
Outbound
1
2
3
4
5
*6
7
*8
*9
*10
*11
*12
*13
*14

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 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

If the commit bit is set by the responder with this proposal, some of the problems described in Section 0 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 makes a proposal for 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) The rules for phase 1 SA 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 SA 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 SA Re-keying Requirements

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

The second reason stated above has been deemed unimportant by the working group as a factor in determining phase 1 SA re-keying 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 implementation will be required to produce a mechanism to tear down SAs created by entities that are no longer authorised.

Originally, this document made the assumption that there must always be a valid phase 1 SA in existence between entities to allow the phase 2 SAs to continue to exist. This was driven primarily by the desire to bound the authorised lifetime of phase 2 SAs by the authorised lifetime of the phase 1 SA. However, as stated above, this requirement has been considered unnecessary by the working group.

Without that requirement, it was decided by the working group that a valid phase 1 SA does not have to exist between two peers that have valid phase 2 SAs.

In spite of this, this document asserts that there is still value in making sure that a valid phase 1 SA exists at all times. This value comes about from the fact that the existence of phase 1 SAs between two entities creates a logical control channel for phase 2 SA management between those entities. This method of phase 1 re-keying will be called the continuous channel method.

The continuous channel method is implicitly recommended by RFC2408. 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.

Unfortunately, the downside of this implementation is that it must be aware of dangling phase 2 SA implementations due to the decision of the working group. This document uses the term "dangling" to describe phase 2 SAs that have no valid phase 1 SA between the two peers. The differences between the two methods are further discussed here.

3.2 Continuous Channel Implementations

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 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.

Therefore, re-keying of phase 1 SAs requires that peers negotiate a new phase 1 SA before the old phase 1 SA expires, at some percentage of the SA's lifetime.

Note that this 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 (if they expire by time). In other words, SAs that are no longer necessary may never disappear.

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, since it is beyond the scope of this document.

The existence of the INITIAL-CONTACT notification determines whether it should delete any phase 2 SAs it has with the peer.

Summarised, the rules for phase 1 re-keying for continuous channel implementations are:

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 complete
-responder deletes all previously existing phase 2 SAs with the peer, if any

Phase 1 SA Ages:
-end point that first detects this negotiates new phase 1 SA; becomes new initiator

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
-responder SHOULD mark its existing phase 1 SA as re-keyed, so as to not re-key again
-since no INITIAL-CONTACT notification is used by either end; phase 2 SAs are kept

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

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 addresses passed using the Configuration Exchange mode.

Also, continuous channel implementations MUST delete all phase 2 SAs with a peer when the last phase 1 SA between them expires. This could happen due to administrative shut-down of the link between the peers, or a policy change that caused the phase 1 SA re-keying to fail, leaving no valid phase 1 SAs between them when the existing phase 1 expired.

A difficulty arises here due to the optional and unacknowledged transmission of the delete notification and the need to be dangling SA aware. If the delete notification for any existing phase 2 SA is dropped, the implementation must attempt to determine if the peer uses the continuous channel method, or is an SA dangler.

A possible state machine for continuous channel implementations is shown in Table 1. Note that this state machine does not cover error

conditions, simultaneous SA negotiations and other events, and is provided for illustration only. It should not be considered a complete design.

# SAs P1	P2	Event	Actions
0	0	-phase 1 negotiation from peer -phase 1 SA needed	-initiate phase 1 negotiation -use initial contact if allowed
1	0	-phase 1 SA negotiation from peer	-respond to phase 1 negotiation -do not use initial contact
		-phase 1 SA deletion from peer	-delete phase 1 SA
		-phase 1 SA ages	-re-key phase 1 SA (or not)
		-phase 1 SA expires	-send delete notification for phase 1 SA -delete phase 1 SA
1	1+	-phase 1 SA negotiation from peer	-respond to phase 1 negotiation -do not use initial contact
		-phase 1 SA ages	-re-key phase 1 SA
		-phase 1 SA deletion from peer	-re-key phase 1 SA (1) -delete phase 1 SA
		-phase 1 SA expires	-send deletes for all phase 2 SAs -delete all phase 2 SAs -send delete notification for phase 1 SA -delete phase 1 SA
2+	1+	-phase 1 SA negotiation from peer	-respond to phase 1 negotiation -do not use initial contact
		-phase 1 SA ages	-re-key the phase 1 SA

		-phase 1 SA deletion from peer	-delete phase 1 SA
		-phase 1 SA expires	-send delete notification for phase 1 SA -delete phase 1 SA
0	1+	-phase 1 negotiation fails (2)	-delete all phase 2 SAs (3) -send no delete notifications

Table 1 Continuous Channel State/Event Table

Table Notes:

(1) This event's actions are a result of dangling SA awareness. In a pure continuous channel system, the action here would be to delete all phase 2 SAs and the phase 1 SA.

(2) This event may result from the action associated with the previous note. Thus, this event is the result of a continuous channel implementation attempting to be dangling SA aware.

(3) Implementations could choose to do nothing here, and behave as an SA dangling implementation. However, if phase 1 SA negotiation fails, in general it means that the control channel between the peers cannot be maintained, suggesting that phase 2 SAs should also be deleted.

3.2.1 Identity Perfect Forward Secrecy

Identity PFS is normally done by allowing the use of only a single quick mode in a phase 1 SA. This is controlled by the deletion of the phase 1 after a quick mode.

In a dangling SA implementation, this is not a problem, since the absence of a valid phase 1 SA is permitted. However, in a continuous channel implementation, this can lead to the absence of the channel.

This is solved by creating two phase 1 SAs before the first quick mode 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 the quick mode, and is immediately deleted.

The phase 1 SA that is assigned to channel management is re-keyed as described here. 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. These other phase 1 SAs are created and used to generate phase 2 SAs, then immediately deleted.

3.3 Dangling Phase 2 SA Implementations

These implementations allow phase 2 SAs to exist without a valid phase 1 SA between peers. They do this by allowing phase 1 SAs to expire without re-keying them, and then re-keying them only when necessary, such as when a phase 2 SA needs re-keying.

Unfortunately, this can lead to situations where phase 2 SAs cannot be properly cleaned up. For example, if an implementation's policy is changed to disallow a user access to a network, all SAs between that user and the network should be terminated. However, if there are no phase 1 SAs between them, none can be re-keyed, so the DELETE notifications for the phase 2 SAs cannot be sent. This can lead to the existence of invalid phase 2 SAs on one end, with the result that encrypted traffic is send from that end that cannot be used. Since phase 1 SAs cannot be created, the SAs cannot be explicitly deleted by the peer.

(If this was a continuous channel implementation, the phase 2 SAs could have been deleted when the change in authorisation was discovered, or when the existing phase 1 SA expired.)

Summarised, the rules for phase 1 re-keying for dangling SA implementations are:

```
  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 complete
-responder deletes all previously existing phase 2 SAs with the peer, if any

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
-responder SHOULD mark all existing phase 1 SAs with same peer as re-keyed, so as to not re-key again
-since no INITIAL-CONTACT notification is used by either end; phase 2 SAs are kept

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

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

A possible state machine for continuous channel implementations is shown in Table 2. Note that this state machine does not cover error conditions, simultaneous SA negotiations and other events, and is provided for illustration only. It should not be considered a complete design.

# SAs P1	# SAs P2	Event	Actions
0	0	-phase 1 negotiation from peer -phase 1 SA needed	-initiate phase 1 negotiation -use initial contact if allowed
1	0	-phase 1 SA negotiation from peer	-respond to phase 1 negotiation -do not use initial contact
		-phase 1 SA deletion from peer	-delete phase 1 SA
		-phase 1 SA expires	-send delete notification for phase 1 SA -delete phase 1 SA
1+	1+	-phase 1 SA negotiation	-respond to phase 1 negotiation

		from peer	-do not use initial contact
		-phase 1 SA deletion from peer	-delete phase 1 SA
		-phase 1 SA expires	-send delete notification for phase 1 SA -delete phase 1 SA
0	1+	-phase 1 SA negotiation from peer	-respond to phase 1 negotiation -do not use initial contact
		-phase 2 SA ages	-attempt to re-negotiate phase 1 SA -re-key phase 2 SA if successful
		-phase 2 SA expires	-attempt to re-negotiate phase 1 SA -send delete notification for phase 2 SA if successful -delete phase 2 SA unconditionally

Table 2 Dangling SA State/Event Table

The differences between Table 1 and Table 2 are the following:

-ageing detection of phase 1 SAs is unnecessary
-the state table is not split based on having 1 or more than 1 phase 1 SA for the event where a phase 1 SA deletion is received
-the state table now depends on phase 2 SA events

While there is more complexity in the continuous channel implementation, it is not excessive, and some of it is required to support the dangling SA implementation.

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 dangles phase 2 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 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 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.

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, implementation 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 responder during aggressive mode is never allowed to use any notification, 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.

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 may not occur at the same time at both ends to 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 better 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 echange. 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 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 describes in detail how the Acknowledged Informational exchange is 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 SA pairs). 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 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.

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 to be deleted 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 IPSecond

The phase 1 re-keying method described in Section 3 requires only one change for IPSecond. That is the required use of the Acknowledged Informational exchange when deleting SAs.

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 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.

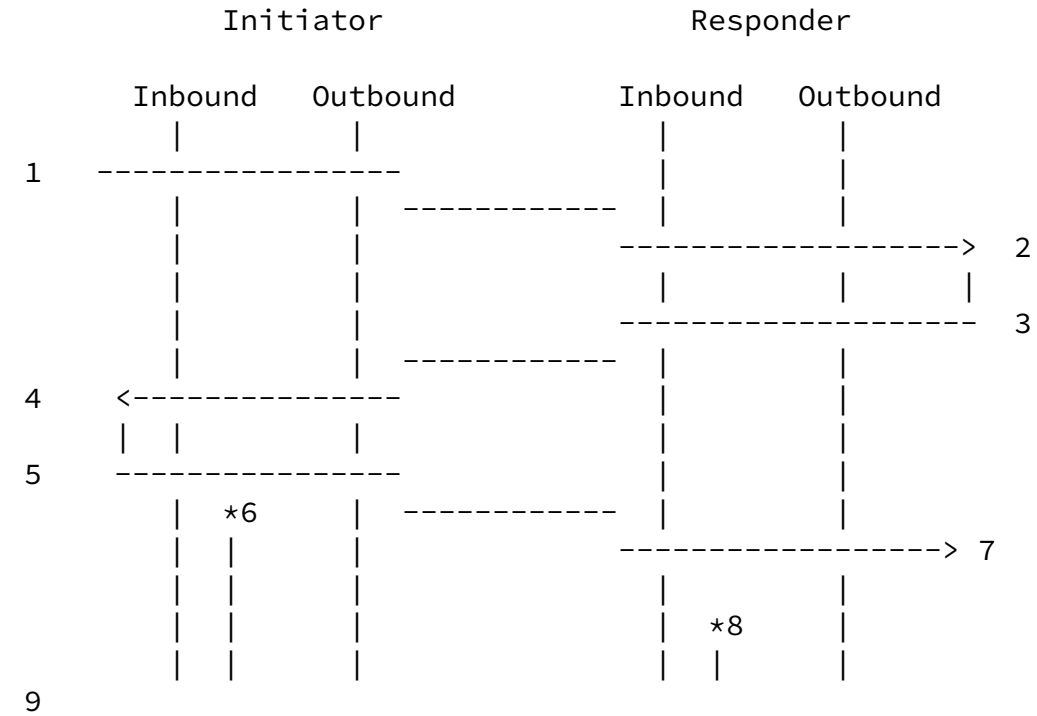
It also eliminates the race condition that occurs during SA set up during re-keying. This means that use of the commit bit to avoid the race condition is not necessary except when the very first phase 2 SA is set up.

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 delete mode 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 and old SA deletion.



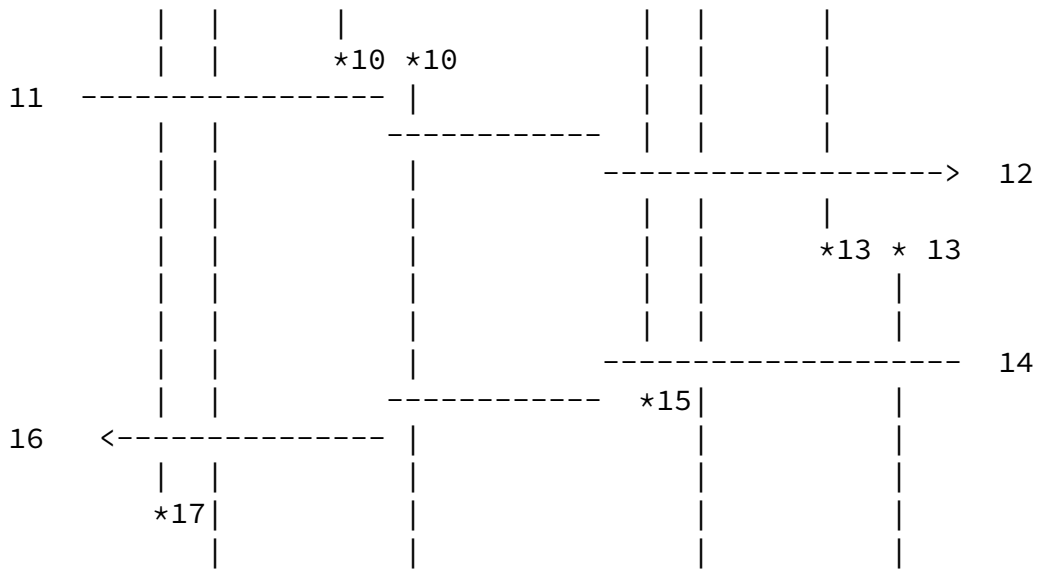


Figure 4-1 Recommended IPSecond Phase 2 Re-key Sequence Chart, Initiator Expiration

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 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.

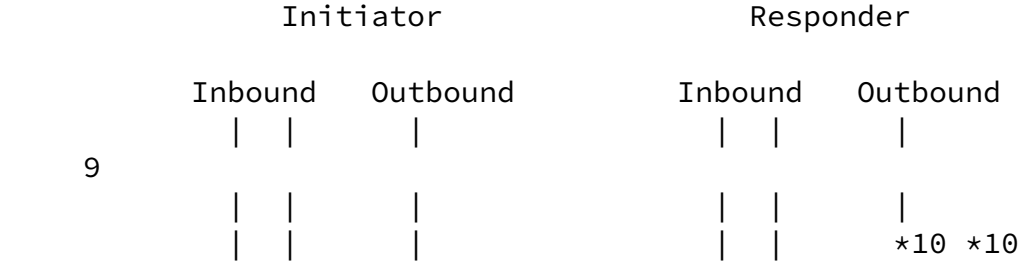
14) Responder sends second Acknowledged Informational exchange message.

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.



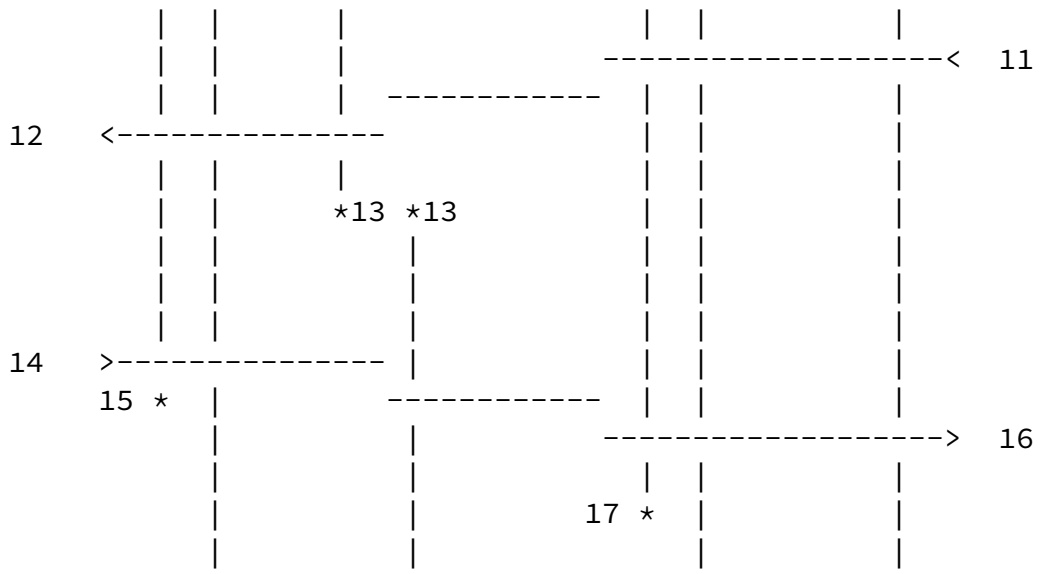


Figure 4-2 Recommended IPSecond Phase 2 Re-key Sequence Chart, Responder Expiration

9) Responder's old SA pair expires.

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

11) Responder sends first 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 introducing 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:

- They receive the full protection of phase 1 SAs, and as such provide the maximum resistance to denial of service attacks.
- Their use is clearly and unambiguously defined.
- 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 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 received 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 mode.

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 use of the SA was deferred by both peers, 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 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.

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

Revision History

September 23, 1998 Initial Release

April 7, 1999 -clarification of objectives of document
-more commit bit comments
-change phase 1 re-keying discussion to explicitly allow multiple phase 1 SAs between peers; previous version explicitly allowed only 1 phase 1 between peers
-other miscellaneous changes

October 20, 1999 -separation of phase 1 re-keying into continuous channel and dangling SA implementations
-use of Acknowledged Information exchange used where possible
-other changes

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@lists.tislabs.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

This document expires April 20, 2000