Network Working Group
Internet Draft
Expires: 26 Aug 2002

Henry Spencer
SP Systems
D. Hugh Redelmeier
Mimosa Systems
26 Feb 2002

IKE Implementation Issues

<draft-spencer-ipsec-ike-implementation-02.txt>

Status of this Memo

Copyright Notice

Table of Contents

Abstract

The current IPsec specifications for key exchange and connection management, RFCs 2408 [ISAKMP] and 2409 [IKE], leave many aspects of connection management unspecified, most prominently rekeying practices. Pending clarifications in future revisions of the specifications, this document sets down some successful experiences, to minimize the extent to which new implementors have to rely on unwritten folklore.

The Linux FreeS/WAN implementation of IPsec interoperates with almost every other IPsec implementation. This document describes how the FreeS/WAN project has resolved some of the gaps in the IPsec specifications (and plans to resolve some others), and what difficulties have been encountered, in hopes that this generally-successful experience might be informative to new implementors.

This is offered as an Informational RFC.

This -02 revision mainly: discusses ISAKMP SA expiry during IPsec-SA rekeying (4.5), revises the discussion of bidirectional Deletes (5.1), suggests remembering the parameters of successful negotiations for later use (4.2, 5.3), notes an unsuccessful negotiation from the other end as a hint of a possibly broken connection (5.5), and adds sections on network partitions (5.4), authentication methods and the subtleties of RSA public keys (6.7), and miscellaneous interoperability concerns (6.8).

1. Introduction

The current IPsec specifications for key exchange and connection management, RFCs 2408 [ISAKMP] and 2409 [IKE], leave many aspects of connection management unspecified, most prominently rekeying practices. This is a cryptic puzzle which each group of implementors has to struggle with, and differences in how the ambiguities and gaps are resolved are potentially a fruitful source of interoperability problems. We can hope that future revisions of the specifications will clear this up. Meanwhile, it seems useful to set down some successful experiences, to minimize the extent to which new implementors have to rely on unwritten folklore.

The Linux FreeS/WAN implementation of IPsec interoperates with almost every other IPsec implementation, and because of its free nature, it also sees some use as a reference implementation by other implementors. The high degree of interoperability is noteworthy given its organizers' strong minimalist bias, which has caused them to implement only a small subset of the full glory of IPsec. This document describes how the FreeS/WAN project has resolved some of the

gaps in the IPsec specifications (and plans to resolve some others), and what difficulties have been encountered, in hopes that this generally-successful experience might be informative to new implementors.

One small caution about applicability: this experience may not be relevant to severely resource-constrained implementations. FreeS/WAN's target environment is previous-generation PCs, now available at trivial cost (often, within an organization, at no cost), which have quite impressive CPU power and memory by the standards of only a few years ago. Some of the approaches discussed here may be inapplicable to implementations with severe external constraints which prevent them from taking advantage of modern hardware technology.

2. Lower-level Background and Notes

2.1. Packet Handling

FreeS/WAN implements ESP [ESP] and AH [AH] straightforwardly, although AH sees little use among our users. Our ESP/AH implementation cannot currently handle packets with IP options; somewhat surprisingly, this has caused little difficulty. We insist on encryption and do not support authentication-only connections, and this has not caused significant difficulty either.

MTU and fragmentation issues, by contrast, have been a constant headache. We will not describe the details of our current approach to them, because it still needs work. One difficulty we have encountered is that many combinations of packet source and packet destination apparently cannot cope with an "interior minimum" in the path MTU, e.g. where an IPsec tunnel intervenes and its headers reduce the MTU for an intermediate link. This is particularly prevalent when using common PC software to connect to large well-known web sites; we think it is largely due to misconfigured firewalls which do not pass ICMP Fragmentation Required messages. The only solution we have yet found is to lie about the MTU of the tunnel, accepting the (undesirable) fragmentation of the ESP packets for the sake of preserving connectivity.

We currently zero out the TOS field of ESP packets, rather than copying it from the inner header, on the grounds that it lends itself too well to traffic analysis and covert channels. We provide an option to restore RFC 2401 [IPSEC] copying behavior, but this appears to see little use.

2.2. Ciphers

We initially implemented both DES [DES] and 3DES [CIPHERS] for both IKE and ESP, but after the Deep Crack effort [CRACK] demonstrated its inherent insecurity, we dropped support for DES. Somewhat surprisingly, our insistence on 3DES has caused almost no interoperability problems, despite DES being officially mandatory. A very few other systems either do not support 3DES or support it only as an optional upgrade, which inconveniences a few would-be users. There have also been one or two cases of systems which don't quite seem to know the difference!

See also section 6.1 for a consequence of our insistence on 3DES.

2.3. Interfaces

We currently employ PF_KEY version 2 [PFKEY], plus various non-standard extensions, as our interface between keying and ESP. This has not proven entirely satisfactory. Our feeling now is that keying issues and policy issues do not really lend themselves to the clean separation that PF_KEY envisions.

3. IKE Infrastructural Issues

A number of problems in IPsec connection management become easier if some attention is first paid to providing an infrastructure to support solving them.

3.1. Continuous Channel

FreeS/WAN uses an approximation to the "continuous channel" model, in which ISAKMP SAs are maintained between IKEs so long as any IPsec SAs are open between the two systems. The resource consumption of this is minor: the only substantial overhead is occasional rekeying. IPsec SA management becomes significantly simpler if there is always a channel for transmission of control messages. We suggest (although we do not yet fully implement this) that inability to maintain (e.g., to rekey) this control path should be grounds for tearing down the IPsec SAs as well.

As a corollary of this, there is one and only one ISAKMP SA maintained between a pair of IKEs (although see sections 5.3 and 6.5 for minor complications).

3.2. Retransmission

The unreliable nature of UDP transmission is a nuisance. IKE implementations should always be prepared to retransmit the most

recent message they sent on an ISAKMP SA, since there is some possibility that the other end did not get it. This means, in particular, that the system sending the supposedly-last message of an exchange cannot relax and assume that the exchange is complete, at least not until a significant timeout has elapsed.

Systems must also retain information about the message most recently received in an exchange, so that a duplicate of it can be detected (and possibly interpreted as a NACK for the response).

The retransmission rules FreeS/WAN follows are: (1) if a reply is expected, retransmit only if it does not appear before a timeout; and (2) if a reply is not expected (last message of the exchange), retransmit only on receiving a retransmission of the previous message. Notably, in case (1) we do NOT retransmit on receiving a retransmission, which avoids possible congestion problems arising from packet duplication, at the price of slowing response to packet loss. The timeout for case (1) is 10 seconds for the first retry, 20 seconds for the second, and 40 seconds for all subsequent retries (normally only one, except when configuration settings call for persistence and the message is the first message of Main Mode with a new peer). These retransmission rules have been entirely successful.

(Michael Thomas of Cisco has pointed out that the retry timeouts should include some random jitter, to de-synchronize hosts which are initially synchronized by, e.g., a power outage. We already jitter our rekeying times, as noted in section 4.2, but that does not help with initial startup. We're implementing jittered retries, but cannot yet report on experience with this.)

There is a deeper problem, of course, when an entire "exchange" consists of a single message, e.g. the ISAKMP Informational Exchange. Then there is no way to decide whether or when a retransmission is warranted at all. This seems like poor design, to put it mildly (and there is now talk of fixing it). We have no experience in dealing with this problem at this time, although it is part of the reason why we have delayed implementing Notification messages.

3.3. Replay Prevention

The unsequenced nature of UDP transmission is also troublesome, because it means that higher levels must consider the possibility of replay attacks. FreeS/WAN takes the position that systematically eliminating this possibility at a low level is strongly preferable to forcing careful consideration of possible impacts at every step of an exchange. RFC 2408 [ISAKMP] section 3.1 states that the Message ID of an ISAKMP message must be "unique". FreeS/WAN interprets this literally, as forbidding duplication of Message IDs within the set of

all messages sent via a single ISAKMP SA.

This requires remembering all Message IDs until the ISAKMP SA is superseded by rekeying, but that is not costly (four bytes per sent or received message), and it ELIMINATES replay attacks from consideration; we believe this investment of resources is well worthwhile. If the resource consumption becomes excessive--in our experience it has not--the ISAKMP SA can be rekeyed early to collect the garbage.

There is theoretically an interoperability problem when talking to implementations which interpret "unique" more loosely and may re-use Message IDs, but it has not been encountered in practice. This approach appears to be completely interoperable.

The proposal by Andrew Krywaniuk [REPLAY], which advocates turning the Message ID into an anti-replay counter, would achieve the same goal without the minor per-message memory overhead. This may be preferable, although it means an actual protocol change and more study is needed.

4. Basic Keying and Rekeying

4.1. When to Create SAs

As Tim Jenkins [REKEY] pointed out, there is a potential race condition in Quick Mode: a fast lightly-loaded Initiator might start using IPsec SAs very shortly after sending QM3 (the third and last message of Quick Mode), while a slow heavily-loaded Responder might not be ready to receive them until after spending a significant amount of time creating its inbound SAs. The problem is even worse if QM3 gets delayed or lost.

FreeS/WAN's approach to this is what Jenkins called "Responder Pre-Setup": the Responder creates its inbound IPsec SAs before it sends QM2, so they are always ready and waiting when the Initiator sends QM3 and begins sending traffic. This approach is simple and reliable, and in our experience it interoperates with everybody. (There is potentially still a problem if FreeS/WAN is the Initiator and the Responder does not use Responder Pre-Setup, but no such problems have been seen.) The only real weakness of Responder Pre-Setup is the possibility of replay attacks, which we have eliminated by other means (see section 3.3).

With this approach, the Commit Bit is useless, and we ignore it. In fact, until quite recently we discarded any IKE message containing it, and this caused surprisingly few interoperability problems; apparently it is not widely used. We have recently been persuaded

that simply ignoring it is preferable; preliminary experience with this indicates that the result is successful interoperation with implementations which set it.

4.2. When to Rekey

To preserve connectivity for user traffic, rekeying of a connection (that is, creation of new IPsec SAs to supersede the current ones) must begin before its current IPsec SAs expire. Preferably one end should predictably start rekeying negotiations first, to avoid the extra overhead of two simultaneous negotiations, although either end should be prepared to rekey if the other does not. There is also a problem with "convoys" of keying negotiations: for example, a "hub" gateway with many IPsec connections can be inundated with rekeying negotiations exactly one connection-expiry time after it reboots, and the massive overload this induces tends to make this situation self-perpetuating, so it recurs regularly. (Convoys can also evolve gradually from initially-unsynchronized negotiations.)

FreeS/WAN has the concept of a "rekeying margin", measured in seconds. If FreeS/WAN was the Initiator for the previous rekeying (or the startup, if none) of the connection, it nominally starts rekeying negotiations at expiry time minus one rekeying margin. Some random jitter is added to break up convoys: rather than starting rekeying exactly at minus one margin, it starts at a random time between minus one margin and minus two margins. (The randomness here need not be cryptographic in quality, so long as it varies over time and between hosts. We use an ordinary PRNG seeded with a few bytes from a cryptographic randomness source. The seeding mostly just ensures that the PRNG sequence is different for different hosts, even if they start up simultaneously.)

If FreeS/WAN was the Responder for the previous rekeying/startup, and nothing has been heard from the previous Initiator at expiry time minus one-half the rekeying margin, FreeS/WAN will initiate rekeying negotiations. No jitter is applied; we now believe that it should be jittered, say between minus one-half margin and minus one-quarter margin.

Having the Initiator lead the way is an obvious way of deciding who should speak first, since there is already an Initiator/Responder asymmetry in the connection. Moreover, our experience has been that Initiator lead gives a significantly higher probability of successful negotiation! The negotiation process itself is asymmetric, because the Initiator must make a few specific proposals which the Responder can only accept or reject, so the Initiator must try to guess where its "acceptable" region (in parameter space) might overlap with the Responder's. We have seen situations where negotiations would

succeed or fail depending on which end initiated them, because one end was making better guesses. Given an existing connection, we KNOW that the previous Initiator WAS able to initiate a successful negotiation, so it should (if at all possible) take the lead again. Also, the Responder should remember the Initiator's successful proposal, and start from that rather than from his own default proposals if he must take the lead; we don't currently implement this completely but plan to.

FreeS/WAN defaults the rekeying margin to 9 minutes, although this can be changed by configuration. There is also a configuration option to alter the permissible range of jitter. The defaults were chosen somewhat arbitrarily, but they work extremely well and the configuration options are rarely used.

4.3. Choosing an SA

Once rekeying has occurred, both old and new IPsec SAs for the connection exist, at least momentarily. FreeS/WAN accepts incoming traffic on either old or new inbound SAs, but sends outgoing traffic only on the new outbound ones. This approach appears to be significantly more robust than using the old ones until they expire, notably in cases where renegotiation has occurred because something has gone wrong on the other end. It avoids having to pay meticulous attention to the state of the other end, state which is difficult to learn reliably given the limitations of IKE.

This approach has interoperated successfully with ALMOST all other implementations. The only (well-characterized) problem cases have been implementations which rely on receiving a Delete message for the old SAs to tell them to switch over to the new ones. Since delivery of Delete is unreliable, and support for Delete is optional, this reliance seems like a serious mistake. This is all the more true because Delete announces that the deletion has already occurred [ISAKMP, section 3.15], not that it is about to occur, so packets already in transit in the other direction could be lost. Delete should be used for resource cleanup, not for switchover control. (These matters are discussed further in section 5.)

4.4. Why to Rekey

FreeS/WAN currently implements only time-based expiry (life in seconds), although we are working toward supporting volume-based expiry (life in kilobytes) as well. The lack of volume-based expiry has not been an interoperability problem so far.

Volume-based expiry does add some minor complications. In particular, it makes explicit Delete of now-disused SAs more

important, because once an SA stops being used, it might not expire on its own. We believe this lacks robustness and is generally unwise, especially given the lack of a reliable Delete, and expect to use volume-based expiry only as a supplement to time-based expiry. However, Delete support (see section 5) does seem advisable for use with volume-based expiry.

We do not believe that volume-based expiry alters the desirability of switching immediately to the new SAs after rekeying. Rekeying margins are normally a small fraction of the total life of an SA, so we feel there is no great need to "use it all up".

4.5. Rekeying ISAKMP SAs

The above discussion has focused on rekeying for IPsec SAs, but FreeS/WAN applies the same approaches to rekeying for ISAKMP SAs, with similar success.

One issue which we have noticed, but not explicitly dealt with, is that difficulties may ensue if an IPsec-SA rekeying negotiation is in progress at the time when the relevant ISAKMP SA gets rekeyed. The IKE specification [IKE] hints, but does not actually say, that a Quick Mode negotiation should remain on a single ISAKMP SA throughout.

A reasonable rekeying margin will generally prevent the old ISAKMP SA from actually expiring during a negotiation. Some attention may be needed to prevent in-progress negotiations from being switched to the new ISAKMP SA. Any attempt at pre-expiry deletion of the ISAKMP SA must be postponed until after such dangling negotiations are completed, and there should be enough delay between ISAKMP-SA rekeying and a deletion attempt to (more or less) ensure that there are no negotiation-starting packets still in transit from before the rekeying.

At present, FreeS/WAN does none of this, and we don't KNOW of any resulting trouble. With normal lifetimes, the problem should be uncommon, and we speculate that an occasional disrupted negotiation simply gets retried.

4.6. Bulk Negotiation

Quick Mode nominally provides for negotiating possibly-large numbers of similar but unrelated IPsec SAs simultaneously [IKE, section 9]. Nobody appears to do this. FreeS/WAN does not support it, and its absence has caused no problems.

5. Deletions, Teardowns, Crashes

FreeS/WAN currently ignores all Notifications and Deletes, and never generates them. This has caused little difficulty in interoperability, which shouldn't be surprising (since Notification and Delete support is officially entirely optional) but does seem to surprise some people. Nevertheless, we do plan some changes to this approach based on past experience.

5.1. Deletions

As hinted at above, we plan to implement Delete support, done as follows. Shortly after rekeying of IPsec SAs, the Responder issues a Delete for its old inbound SAs (but does not actually delete them yet). The Responder initiates this because the Initiator started using the new SAs on sending QM3, while the Responder started using them only on (or somewhat after) receiving QM3, so there is less chance of old-SA packets still being in transit from the Initiator. The Initiator issues an unsolicited Delete only if it does not hear one from the Responder after a longer delay.

Either party, on receiving a Delete for one or more of the old outbound SAs of a connection, deletes ALL the connection's SAs, and acknowledges with a Delete for the old inbound SAs. A Delete for nonexistent SAs (e.g., SAs which have already been expired or deleted) is ignored. There is no retransmission of unacknowledged Deletes.

In the normal case, with prompt reliable transmission (except possibly for loss of the Responder's initial Delete) and conforming implementations on both ends, this results in three Deletes being transmitted, resembling the classic three-way handshake. Loss of a Delete after the first, or multiple losses, will cause the SAs not to be deleted on at least one end. It appears difficult to do much better without at least a distinction between request and acknowledgement.

RFC 2409 section 9 "strongly suggests" that there be no response to informational messages such as Deletes, but the only rationale offered is prevention of infinite loops endlessly exchanging "I don't understand you" informationals. Since Deletes cannot lead to such a loop (and in any case, the nonexistent-SA rule prevents more than one acknowledgement for the same connection), we believe this recommendation is inapplicable here.

As noted in section 4.3, these Deletes are intended for resource cleanup, not to control switching between SAs. But we expect that they will improve interoperability with some broken implementations.

We believe strongly that connections need to be considered as a whole, rather than treating each SA as an independent entity. We will issue Deletes only for the full set of inbound SAs of a connection, and will treat a Delete for any outbound SA as equivalent to deletion of all the outbound SAs for the associated connection.

The above is phrased in terms of IPsec SAs, but essentially the same approach can be applied to ISAKMP SAs (the Deletes for the old ISAKMP SA should be sent via the new one).

5.2. Teardowns and Shutdowns

When a connection is not intended to be up permanently, there is a need to coordinate teardown, so that both ends are aware that the connection is down. This is both for recovery of resources, and to avoid routing packets through dangling SAs which can no longer deliver them.

Connection teardown will use the same bidirectional exchange of Deletes as discussed in section 5.1: a Delete received for current IPsec SAs (not yet obsoleted by rekeying) indicates that the other host wishes to tear down the associated connection.

A Delete received for a current ISAKMP SA indicates that the other host wishes to tear down not only the ISAKMP SA but also all IPsec SAs currently under the supervision of that ISAKMP SA. The 5.1 bidirectional exchange might seem impossible in this case, since reception of an ISAKMP-SA Delete indicates that the other end will ignore further traffic on that ISAKMP SA. We suggest using the same tactic discussed in 5.1 for IPsec SAs: the first Delete is sent without actually doing the deletion, and the response to receiving a Delete is to do the deletion and reply with another Delete. If there is no response to the first Delete, retry a small number of times and then give up and do the deletion; apart from being robust against packet loss, this also maximizes the probability that an implementation which does not do the bidirectional Delete will receive at least one of the Deletes.

When a host with current connections knows that it is about to shut down, it will issue Deletes for all SAs involved (both IPsec and ISAKMP), advising its peers (as per the meaning of Delete [ISAKMP, section 3.15]) that the SAs have become useless. It will ignore attempts at rekeying or connection startup thereafter, until it shuts down.

It would be better to have a Final-Contact notification, analogous to Initial-Contact but indicating that no new negotiations should be attempted until further notice. Initial-Contact actually could be

used for shutdown notification (!), but in networks where connections are intended to exist permanently, it seems likely to provoke unwanted attempts to renegotiate the lost connections.

5.3. Crashes

Systems sometimes crash. Coping with the resulting loss of information is easily the most difficult problem we have found in implementing robust IPsec systems.

When connections are intended to be permanent, it is simple to specify renegotiation on reboot. With our approach to SA selection (see section 4.3), this handles such cases robustly and well. We do have to tell users that BOTH hosts should be set this way. In cases where crashes are synchronized (e.g. by power interruptions), this may result in simultaneous negotiations at reboot. We currently allow both negotiations to proceed to completion, but our use-newest selection method effectively ignores one connection or the other, and when one of them rekeys, we notice that the new SAs replace those of both old connections, and we then refrain from rekeying the other. (This duplicate detection is desirable in any event, for robustness, to ensure that the system converges on a reasonable state eventually after it is perturbed by difficulties or bugs.)

When connections are not permanent, the situation is less happy. One particular situation in which we see problems is when a number of "Road Warrior" hosts occasionally call in to a central server. The server is normally configured not to initiate such connections, since it does not know when the Road Warrior is available (or what IP address it is using). Unfortunately, if the server crashes and reboots, any Road Warriors then connected have a problem: they don't know that the server has crashed, so they can't renegotiate, and the server has forgotten both the connections and their (transient) IP addresses, so it cannot renegotiate.

We believe that the simplest answer to this problem is what John Denker has dubbed "address inertia": the server makes a best-effort attempt to remember (in nonvolatile storage) which connections were active and what the far-end addresses were (and what the successful proposal's parameters were), so that it can attempt renegotiation on reboot. We have not implemented this yet, but intend to; Denker has implemented it himself, although in a somewhat messy way, and reports excellent results.

5.4. Network Partitions

A network partition, making the two ends unable to reach each other, has many of the same characteristics as having the other end crash...

until the network reconnects. It is desirable that recovery from this be automatic.

If the network reconnects before any rekeying attempts or other IKE activities occurred, recovery is fully transparent, because the IKEs have no idea that there was any problem. (Complaints such as ICMP Host Unreachable messages are unauthenticated and hence cannot be given much weight.) This fits the general mold of TCP/IP: if nobody wanted to send any traffic, a network outage doesn't matter.

If IKE activity did occur, the IKE implementation will discover that the other end doesn't seem to be responding. The preferred response to this depends on the nature of the connection. If it was intended to be ephemeral (e.g. opportunistic encryption [OE]), closing it down after a few retries is reasonable. If the other end is expected to sometimes drop the connection without warning, it may not be desirable to retry at all. (We support both these forms of configurability, and indeed we also have a configuration option to suppress rekeying entirely on one end.)

If the connection was intended to be permanent, however, then persistent attempts to re-establish it are appropriate. Some degree of backoff is appropriate here, so that retries get less frequent as the outage gets prolonged. Backoff should be limited, so that re-established connectivity is not followed by a long delay before a retry. Finally, after many retries (say 24 hours' worth), it may be preferable to just declare the connection down and rely on manual intervention to re-establish it, should this be desirable. We do not yet fully support all this.

5.5. Unknown SAs

A more complete solution to crashes would be for an IPsec host to note the arrival of ESP packets on an unknown IPsec SA, and report it somehow to the other host, which can then decide to renegotiate. This arguably might be preferable in any case--if the non-rebooted host has no traffic to send, it does not care whether the connection is intact--but delays and packet loss will be reduced if the connection is renegotiated BEFORE there is traffic for it. So unknown-SA detection is best reserved as a fallback method, with address inertia used to deal with most such cases.

A difficulty with unknown-SA detection is, just HOW should the other host be notified? IKE provides no good way to do the notification: Notification payloads (e.g., Initial-Contact) are unauthenticated unless they are sent under protection of an ISAKMP SA. A "Security Failures - Bad SPI" ICMP message [SECFAIL] is an interesting alternative, but has the disadvantage of likewise being

unauthenticated. It's fundamentally unlikely that there is a simple solution to this, given that almost any way of arranging or checking authentication for such a notification is costly.

We think the best answer to this is a two-step approach. An unauthenticated Initial-Contact or Security Failures - Bad SPI cannot be taken as a reliable report of a problem, but can be taken as a hint that a problem MIGHT exist. Then there needs to be some reliable way of checking such hints, subject to rate limiting since the checks are likely to be costly (and checking the same connection repeatedly at short intervals is unlikely to be worthwhile anyway). So the rebooted host sends the notification, and the non-rebooted host--which still thinks it has a connection--checks whether the connection still works, and renegotiates if not.

Also, if an IPsec host which believes it has a connection to another host sees an unsuccessful attempt by that host to negotiate a new one, that is also a hint of possible problems, justifying a check and possible renegotiation. ("Unsuccessful" here means a negotiation failure due to lack of a satisfactory proposal. A failure due to authentication failure suggests a denial-of-service attack by a third party, rather than a genuine problem on the legitimate other end.) As noted in section 4.2, it is possible for negotiations to succeed or fail based on which end initiates them, and some robustness against that is desirable.

We have not yet decided what form the notification should take. IKE Initial-Contact is an obvious possibility, but has some disadvantages. It does not specify which connection has had difficulties. Also, the specification [IKE section 4.6.3.3] refers to "remote system" and "sending system" without clearly specifying just what "system" means; in the case of a multi-homed host using multiple forms of identification, the question is not trivial. Initial-Contact does have the fairly-decisive advantage that it is likely to convey the right general meaning even to an implementation which does not do things exactly the way ours does.

A more fundamental difficulty is what form the reliable check takes. What is wanted is an "IKE ping", verifying that the ISAKMP SA is still intact (it being unlikely that IPsec SAs have been lost while the ISAKMP SA has not). The lack of such a facility is a serious failing of IKE. An acknowledged Notification of some sort would be ideal, but there is none at present. Some existing implementations are known to use the private Notification values 30000 as ping and 30002 as ping reply, and that seems the most attractive choice at present. If it is not recognized, there will probably be no reply, and the result will be an unnecessary renegotiation, so this needs strict rate limiting. (Also, when a new connection is set up, it's

probably worth determining by experiment whether the other end supports IKE ping, and remembering that.)

While we think this facility is desirable, and is about the best that can be done with the poor tools available, we have not gotten very far in implementation and cannot comment intelligently about how well it works or interoperates.

6. Misc. IKE Issues

6.1. Groups 1 and 5

We have dropped support for the first Oakley Group (group 1), despite it being officially mandatory, on the grounds that it is grossly too weak to provide enough randomness for 3DES. There have been some interoperability problems, mostly quite minor: ALMOST everyone supports group 2 as well, although sometimes it has to be explicitly configured.

We also support the quasi-standard group 5 [GROUPS]. This has not been seriously exercised yet, because historically we offered group 2 first and almost everyone accepted it. We have recently changed to offering group 5 first, and no difficulties have been reported.

6.2. To PFS Or Not To PFS

A persistent small interoperability problem is that the presence or absence of PFS (for keys [IKE, section 5.5]) is neither negotiated nor announced. We have it enabled by default, and successful interoperation often requires having the other end turn it on in their implementation, or having the FreeS/WAN end disable it. Almost everyone supports it, but it's usually not the default, and interoperability is often impossible unless the two ends somehow reach prior agreement on it.

We do not explicitly support the other flavor of PFS, for identities [IKE, section 8], and this has caused no interoperability problems.

6.3. Debugging Tools, Lack Thereof

We find IKE lacking in basic debugging tools. Section 5.4, above, notes that an IKE ping would be useful for connectivity verification. It would also be extremely helpful for determining that UDP/500 packets get back and forth successfully between the two ends, which is often an important first step in debugging.

It's also quite common to have IKE negotiate a connection successfully, but to have some firewall along the way blocking ESP.

Users find this mysterious and difficult to diagnose. We have no immediate suggestions on what could be done about it.

6.4. Terminology, Vagueness Thereof

The terminology of IPsec needs work. We feel that both the specifications and user-oriented documentation would be greatly clarified by concise, intelligible names for certain concepts.

We semi-consistently use "group" for the set of IPsec SAs which are established in one direction by a single Quick Mode negotiation and are used together to process a packet (e.g., an ESP SA plus an AH SA), "connection" for the logical packet path provided by a succession of pairs of groups (each rekeying providing a new pair, one group in each direction), and "keying channel" for the corresponding supervisory path provided by a sequence of ISAKMP SAs.

We think it's a botch that "PFS" is used to refer to two very different things, but we have no specific new terms to suggest, since we only implement one kind of PFS and thus can just ignore the other.

6.5. A Question of Identity

One specification problem deserves note: exactly when can an existing phase 1 negotiation be re-used for a new phase 2 negotiation, as IKE [IKE, section 4] specifies? Presumably, when it connects the same two "parties"... but exactly what is a "party"?

As noted in section 5.4, in cases involving multi-homing and multiple identities, it's not clear exactly what criteria are used for deciding whether the intended far end for a new negotiation is the same one as for a previous negotiation. Is it by Identification Payload? By IP address? Or what?

We currently use a somewhat-vague notion of "identity", basically what gets sent in Identification Payloads, for this, and this seems to be successful, but we think this needs better specification.

6.6. Opportunistic Encryption

Further IKE challenges appear in the context of Opportunistic Encryption [OE], but operational experience with it is too limited as yet for us to comment usefully right now.

6.7. Authentication and RSA Keys

We provide two IKE authentication methods: shared secrets ("pre-shared keys") and RSA digital signatures. (A user-provided add-on

package generalizes the latter to limited support for certificates; we have not worked extensively with it ourselves yet and cannot comment on it yet.)

Shared secrets, despite their administrative difficulties, see considerable use, and are also the method of last resort for interoperability problems.

For digital signatures, we have taken the somewhat unorthodox approach of using "bare" RSA public keys, either supplied in configuration files or fetched from DNS, rather than getting involved in the complexity of certificates. We encode our RSA public keys using the DNS KEY encoding [DNSRSA] (aka "RFC 2537", although that RFC is now outdated), which has given us no difficulties and which we highly recommend. We have seen two difficulties in connection with RSA keys, however.

First, while a number of IPsec implementations are able to take "bare" RSA public keys, each one seems to have its own idea of what format should be used for transporting them. We've had little success with interoperability here, mostly because of key-format issues; the implementations generally WILL interoperate successfully if you can somehow get an RSA key into them at all, but that's hard. X.509 certificates seem to be the lowest (!) common denominator for key transfer.

Second, although the content of RSA public keys has been stable, there has been a small but subtle change over time in the content of RSA private keys. The "internal modulus", used to compute the private exponent "d" from the public exponent "e" (or vice-versa) was originally [RSA] [PKCS1v1] [SCHNEIER] specified to be (p-1)*(q-1), where p and q are the two primes. However, more recent definitions [PKCS1v2] call it "lambda(n)" and define it to be lcm(p-1, q-1); this appears to be a minor optimization. The result is that private keys generated with the new definition often fail consistency checks in implementations using the old definition. Fortunately, it is seldom necessary to move private keys around. Our software now consistently uses the new definition (and thus will accept keys generated with either definition), but our key generator also has an option to generate old-definition keys, for the benefit of users who upgrade their networks incrementally.

6.8. Misc. Snags

Nonce size is another characteristic that is neither negotiated nor announced but that the two ends must somehow be able to agree on. Our software accepts anything between 8 and 256, and defaults to 16. These numbers were chosen rather arbitrarily, but we have seen no

interoperability failures here.

Nothing in the ISAKMP [ISAKMP] or IKE [IKE] specifications says explicitly that a normal Message ID must be non-zero, but a zero Message ID in fact causes failures.

Similarly, there is nothing in the specs which says that ISAKMP cookies must be non-zero, but zero cookies will in fact cause trouble.

7. Security Considerations

Since this document discusses aspects of building robust and interoperable IPsec implementations, security considerations permeate it.

8. References

[AH] Kent, S., and Atkinson, R., "IP Authentication Header", RFC 2402, Nov 1998.

[CIPHERS] Pereira, R., and Adams, R., "The ESP CBC-Mode Cipher Algorithms", RFC 2451, Nov 1998.

[CRACK] Electronic Frontier Foundation, "Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design", O'Reilly 1998, ISBN 1-56592-520-3.

[DES] Madson, C., and Doraswamy, N., "The ESP DES-CBC Cipher Algorithm", RFC 2405, Nov 1998.

[DNSRSA] D. Eastlake 3rd, "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)", RFC 3110, May 2001.

[ESP] Kent, S., and Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC 2406, Nov 1998.

[GROUPS] Kivinen, T., and Kojo, M., "More MODP Diffie-Hellman groups for IKE", <draft-ietf-ipsec-ike-modp-groups-04.txt>, 13 Dec 2001 (work in progress).

[IKE] Harkins, D., and Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, Nov 1998.

[IPSEC] Kent, S., and Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, Nov 1998.

[ISAKMP] Maughan, D., Schertler, M., Schneider, M., and Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, Nov 1998.

[OE] Richardson, M., Redelmeier, D. H., and Spencer, H., "A method for doing opportunistic encryption with IKE", <draft-richardson-ipsec-opportunistic-06.txt>, 21 Feb 2002 (work in progress).

[PKCS1v1] Kaliski, B., "PKCS #1: RSA Encryption, Version 1.5", RFC 2313, March 1998.

[PKCS1v2] Kaliski, B., and Staddon, J., "PKCS #1: RSA Cryptography Specifications, Version 2.0", RFC 2437, Oct 1998.

[PFKEY] McDonald, D., Metz, C., and Phan, B., "PF_KEY Key Management API, Version 2", RFC 2367, July 1998.

[REKEY] Tim Jenkins, "IPsec Re-keying Issues", <draft-jenkins-ipsec-rekeying-06.txt>, 2 May 2000 (draft expired, work no longer in progress).

[REPLAY] Krywaniuk, A., "Using Isakmp Message Ids for Replay Protection", <draft-krywaniuk-ipsec-antireplay-00.txt>, 9 July 2001 (work in progress).

[RSA] Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM v21n2, Feb 1978, p. 120.

[SCHNEIER] Bruce Schneier, "Applied Cryptography", 2nd ed., Wiley 1996, ISBN 0-471-11709-9.

[SECFAIL] Karn, P., and Simpson, W., "ICMP Security Failures Messages", RFC 2521, March 1999.

Authors' Addresses

Henry Spencer
SP Systems
Box 280 Stn. A
Toronto, Ont. M5W1B2
Canada

henry@spsystems.net
416-690-6561

D. Hugh Redelmeier
Mimosa Systems Inc.
29 Donino Ave.
Toronto, Ont. M4N2W6
Canada

hugh@mimosa.com
416-482-8253

Full Copyright Statement