IKEv2 Clarifications and Implementation Guidelines  
draft-eronen-ipsec-ikev2-clarifications-02.txt

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

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 obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

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.

This Internet-Draft will expire on September 25, 2005.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document clarifies many areas of the IKEv2 specification. It does not to introduce any changes to the protocol, but rather provides descriptions that are less prone to ambiguous interpretations. The purpose of this document is to encourage the development of interoperable implementations.
Table of Contents

1. Introduction ............................................ 4
2. Authentication .......................................... 4
   2.1 Data included in AUTH payload calculation ............ 4
   2.2 Hash function for RSA signatures ...................... 5
   2.3 Encoding method for RSA signatures .................... 6
   2.4 Identification type for EAP ........................... 6
   2.5 Identity for policy lookups when using EAP .......... 7
   2.6 EAP authentication and the AUTH payload .......... 7
   2.7 Certificate encoding types ........................... 7
   2.8 Shared key authentication and fixed PRF key size ... 8
   2.9 EAP authentication and fixed PRF key size ........... 9
3. Keying and rekeying ....................................... 9
   3.1 Semantics of the CREATE_CHILD_SA exchange .......... 9
   3.2 Rekeying the IKE_SA vs. reauthentication .......... 13
   3.3 SPIs when rekeying the IKE_SA ....................... 13
   3.4 SPI when rekeying a CHILD_SA ....................... 14
   3.5 Deleting SAs ...................................... 14
4. Traffic selectors ........................................ 14
   4.1 Semantics of complex traffic selector payloads ...... 14
   4.2 ICMP type/code in traffic selector payloads .......... 15
   4.3 Mobility header in traffic selector payloads .......... 15
   4.4 Narrowing the traffic selectors ..................... 15
   4.5 SINGLE_PAIR_REQUIRED ............................. 16
   4.6 Traffic selectors violating own policy ............. 16
5. Configuration payloads ..................................... 17
   5.1 Length of configuration attribute type field ...... 17
   5.2 Requesting any INTERNAL_IP4/IP6_ADDRESS .......... 17
   5.3 INTERNAL_IP4_SUBNET/INTERNAL_IP6_SUBNET .......... 18
   5.4 INTERNAL_IP4_NETMASK ............................... 20
   5.5 Configuration payloads for IPv6 .................... 22
   5.6 INTERNAL_IP6_ADDRESS prefix length ................ 22
   5.7 INTERNAL_IP6_NBNS .................................. 23
6. Miscellaneous issues ...................................... 23
   6.1 Diffie-Hellman for first CHILD_SA .................. 24
   6.2 Extended Sequence Numbers (ESN) transform .......... 24
   6.3 Matching ID_IPV4_ADDR and ID_IPV6_ADDR .............. 24
   6.4 Relationship of IKEv2 to RFC2401bis ............... 25
   6.5 Reducing the window size ............................ 25
   6.6 Minimum size of nonces ............................. 25
   6.7 Initial zero octets on port 4500 .................... 25
   6.8 SPI values in IKE_SA_INIT exchange ................. 26
   6.9 SPI values for messages outside of an IKE_SA ....... 27
   6.10 Protocol ID/SPI fields in Notify payloads .......... 27
   6.11 INVALID_IKE_SPI .................................. 27
   6.12 Which message should contain INITIAL_CONTACT .... 28
7. Status of the clarifications ............................... 28
1. Introduction

This document clarifies many areas of the IKEv2 specification that may be difficult to understand to developers not intimately familiar with the specification and its history. The clarifications in this document come from the discussion on the IPsec WG mailing list, from experience in interoperability testing, and from implementation issues that have been brought to the editors’ attention.

Readers are advised that this document is work-in-progress, and may contain incorrect interpretations. Issues where the clarification is known to be incomplete, or there is no consensus on what the interpretation should be, are marked as such.

IKEv2/IPsec can be used for several different purposes, including IPsec-based remote access (sometimes called the "road warrior" case), site-to-site virtual private networks (VPNs), and host-to-host protection of application traffic. While this document attempts to consider all of these uses, the remote access scenario has perhaps received more attention here than the other uses.

This document does not place any requirements on anyone, and does not use [RFC2119] keywords such as "MUST" and "SHOULD", except in quotations from the original IKEv2 documents. The requirements are given in the IKEv2 specification [IKEv2] and IKEv2 cryptographic algorithms document [IKEv2ALG].

In this document, references to a numbered section (such as "Section 2.15") mean that section in [IKEv2]. References to mailing list messages refer to the IPsec WG mailing list at ipsec@ietf.org. Archives of the mailing list can be found at <http://www.ietf.org/mail-archive/web/ipsec/index.html>.

2. Authentication

2.1 Data included in AUTH payload calculation

Section 2.15 describes how the AUTH payloads are calculated; this calculation involves values prf(SK_pi,Idi’) and prf(SK_pr,Idr’). The text describes the method in words, but does not give clear definitions of what is signed or MACed.

The initiator’s signed octets can be described as:
InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr | ... | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)

The responder’s signed octets can be described as:

ResponderSignedOctets = RealMessage2 | NonceIData | MACedIDForR
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr | ... | Length
RealMessage2 = RealIKEHDR | RestOfMessage2
NonceIPayload = PayloadHeader | NonceIData
ResponderIDPayload = PayloadHeader | RestOfIDPayload
RestOfRespIDPayload = IDType | RESERVED | InitIDData
MACedIDForR = prf(SK_pr, RestOfRespIDPayload)

2.2 Hash function for RSA signatures

Section 3.8 says that RSA digital signature is "Computed as specified in section 2.15 using an RSA private key over a PKCS#1 padded hash."

Unlike IKEv1, IKEv2 does not negotiate a hash function for the IKE_SA. The algorithm for signatures is selected by the signing party who, in general, may not know beforehand what algorithms the verifying party supports. Furthermore, [IKEv2ALG] does not say what algorithms implementations are required or recommended to support. This clearly has a potential for causing interoperability problems, since authentication will fail if the signing party selects an algorithm that is not supported by the verifying party, or not acceptable according to the verifying party’s policy.

This document recommends that all implementations support SHA-1, and use SHA-1 as the default hash function when generating the signatures, unless there are good reasons (such as explicit manual configuration) to believe that the other end supports something else.

Note that unlike IKEv1, IKEv2 uses the PKCS#1 v1.5 [PKCS1v20] signature encoding method (see next section for details), which includes the algorithm identifier for the hash algorithm. Thus, when the verifying party receives the AUTH payload it can determine which hash function was used.

Other possible choices include, for example, using the hash function...
that was used to sign the certificate. However, this approach assumes that the recipient's "IKEv2 module" supports the same algorithms as the "certificate validation module" (which may not be true, especially if something like [SCVP] is used). Furthermore, not all CERT payloads types include a signature; and the certificate could be signed with some other algorithm than RSA.

(References: Magnus Alstrom’s mail "RE:", 2005-01-03. Pasi Eronen’s reply, 2005-01-04. Tero Kivinen’s reply, 2005-01-04.)

2.3 Encoding method for RSA signatures

Section 3.8 says that the RSA digital signature is "Computed as specified in section 2.15 using an RSA private key over a PKCS#1 padded hash."

The current version of PKCS#1 (v2.1) [PKCS1v21] defines two different encoding methods (ways of "padding the hash") for signatures. However, IKEv2 points to the older PKCS#1 v2.0 [PKCS1v20]. That version has only one encoding method for signatures (EMSA-PKCS1-v1_5), and thus there is no ambiguity.

Note that this encoding method is different from the encoding method used in IKEv1. If future revisions of IKEv2 provide support for other encoding methods (such as EMSA-PSS), they will be given new Auth Method numbers.

(References: Pasi Eronen’s mail "RE:", 2005-01-04.)

2.4 Identification type for EAP

Section 3.5 defines several different types for identification payloads, including, e.g., ID_FQDN, ID_RFC822_ADDR, and ID_KEY_ID. EAP [EAP] does not mandate the use of any particular type of identifier, but often EAP is used with Network Access Identifiers (NAIs) defined in [NAI] and [NAIbis]. Although NAIs look a bit like email addresses (e.g., "joe@example.com"), the syntax is not exactly the same as the syntax of email address in [RFC822]. This raises the question of which identification type should be used.

This document recommends that ID_RFC822_ADDR identification type is used for those NAIs that include the realm component. Therefore, responder implementations should not attempt to verify that the contents actually conform to the exact syntax given in [RFC822] or [RFC2822], but instead should accept any reasonable looking NAI.

For NAIs that do not include the realm component, this document recommends using the ID_KEY_ID identification type.
2.5 Identity for policy lookups when using EAP

When the initiator authentication uses EAP, it is possible that the contents of the IDi payload is used only for AAA routing purposes and selecting which EAP method to use. This value may be different from the identity authenticated by the EAP method (see [EAP], Sections 5.1 and 7.3).

It is important that policy lookups and access control decisions use the actual authenticated identity. Often the EAP server is implemented in a separate AAA server that communicates with the IKEv2 responder using, e.g., RADIUS [RADEAP]. In this case, the authenticated identity has to be sent from the AAA server to the IKEv2 responder.


2.6 EAP authentication and the AUTH payload

Section 2.16 says that "For EAP methods that create a shared key as a side effect of authentication, that shared key MUST be used by both the initiator and responder to generate AUTH payloads in messages 5 and 6 using the syntax for shared secrets specified in section 2.15."

This text should say "messages 7 and 8".

(References: "How to do authentication with EAP" thread, Feb 2005)

2.7 Certificate encoding types

Section 3.6 defines a total of twelve different certificate encoding types, and continues that "Specific syntax is for some of the certificate type codes above is not defined in this document." However, the text does not provide references to other documents that would contain information about the exact contents and use of those values.
Without this information, it is not possible to develop interoperable implementations. Therefore, this document recommends that the following certificate encoding values should not be used before new specifications that specify their use are available.

- PKCS #7 wrapped X.509 certificate (1)
- PGP Certificate (2)
- DNS Signed Key (3)
- Kerberos Token (6)
- SPKI Certificate (9)

(Future versions of this document may also contain clarifications about how these values are to be used.)

This document recommends that most implementations should use only those values that are "MUST"/"SHOULD" requirements in [IKEv2]; i.e., "X.509 Certificate - Signature" (4), "Raw RSA Key" (11), "Hash and URL of X.509 certificate" (12), and "Hash and URL of X.509 bundle" (13).

Furthermore, Section 3.7 says that the "Certificate Encoding" field for the Certificate Request payload uses the same values as for Certificate payload. However, the contents of the "Certification Authority" field are defined only for X.509 certificates (presumably covering at least types 4, 10, 12, and 13). This document recommends that other values should not be used before new specifications that specify their use are available.

### 2.8 Shared key authentication and fixed PRF key size

Section 2.15 says that "If the negotiated prf takes a fixed size key, the shared secret MUST be of that fixed size". This statement is correct: the shared secret must be of the correct size. If it is not, it cannot be used; there is no padding, truncation, or other processing involved to force it to that correct size.

This requirement means that it is difficult to use these PRFs with shared key authentication. The authors think this part of the specification was very poorly thought out, and using PRFs with a fixed key size is likely to result in interoperability problems. Thus, we recommend that such PRFs (currently only PRF_AES128_CBC) should not be used with shared key authentication.

Note that Section 2.13 also contains text that is related to PRFs with fixed key size: "When the key for the prf function has fixed length, the data provided as a key is truncated or padded with zeros as necessary unless exceptional processing is explained following the formula". However, this text applies only to the prf+ construction,
so it does not contradict the text in Section 2.15.


2.9 EAP authentication and fixed PRF key size

As described in the previous section, PRFs with a fixed key size require a shared secret of exactly that size. A strict interpretation of this text also means that such PRFs are unlikely to be useful for EAP authentication, since [EAP] specifies that the MSK is at least 64 octets (512 bits) long, while PRF_AES128_CBC requires a 128-bit key. It is currently under discussion whether truncation or padding should be allowed in the EAP case (where the security implications of truncation are slightly different).

(References: Thread "Question about PRFs with fixed size key", Jan 2005.)

3. Keying and rekeying

3.1 Semantics of the CREATE_CHILD_SA exchange

Section 1.3’s organization does not lead to clear understanding of what is needed in which environment. The section can be reorganized with subsections for each use of the CREATE_CHILD_SA exchange (creating child SAs, rekeying IKE SAs, and rekeying child SAs.)

Further, specific parts of Section 3.1 can be clarified. These include:

- It is not clear which SA to send in a rekeying a child SA. The relevant sentence says "If this CREATE_CHILD_SA exchange is rekeying an existing SA other than the IKE_SA, the leading N payload of type REKEY_SA MUST identify the SA being rekeyed." That can be clarified by adding "sender’s inbound" before "SA being rekeyed".

- The specific method for rekeying an IKE_SA is not described in the section that describes the rekeying. This is described in Section 2.8. Relevant text from Section 2.8 can be moved here.

- Section 1.3 never mentions the REKEY_SA Notification, but it does have a mandatory Notification payload when rekeying. The CREATE_CHILD_SA exchange MUST include a REKEY_SA Notification payload with an SPI field identifying the SA being rekeyed.
The spec is partially wrong about the use of nonces in computing keys for CHILD_SAs. Section 1.3 says "The nonces from the initial exchange are used in computing the keys for the CHILD_SA." However, that is not always true. It is only true for a CHILD_SA created in the IKE_AUTH exchange. Thus, the sentence can be ignored because the use of the nonces for computing the keys is clear in Section 2.17.

The new Section 1.3 with subsections and the above changes might look like this.

NEW-1.3 The CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA Exchange is used to create new CHILD_SAs and to rekey both IKE_SAs and CHILD_SAs. This exchange consists of a single request/response pair, and some of its function was referred to as a phase 2 exchange in IKEv1. It MAY be initiated by either end of the IKE-SA after the initial exchanges are completed.

All messages following the initial exchange are cryptographically protected using the cryptographic algorithms and keys negotiated in the first two messages of the IKE exchange. These subsequent messages use the syntax of the Encrypted Payload described in section 3.14. All subsequent messages included an Encrypted Payload, even if they are referred to in the text as "empty". For both messages in the CREATE_CHILD_SA, the message following the header is encrypted and the message including the header is integrity protected using the cryptographic algorithms negotiated for the IKE_SA.

The CREATE_CHILD_SA is used for rekeying IKE_SAs and CHILD_SAs. This section describes the first part of rekeying, the creation of new SAs; Section 2.8 covers the mechanics of rekeying, including moving traffic from old to new SAs and the deletion of the old SAs. The two sections must be read together to understand the entire process of rekeying.

Either endpoint may initiate a CREATE_CHILD_SA exchange, so in this section the term initiator refers to the endpoint initiating this exchange. An implementation MAY refuse all CREATE_CHILD_SA requests within an IKE_SA.

The CREATE_CHILD_SA request MAY optionally contain a KE payload for an additional Diffie-Hellman exchange to enable stronger guarantees of forward secrecy for the CHILD_SA. The keying material for the CHILD_SA is a function of SK_d established during the establishment of the IKE_SA, the nonces exchanged...
during the CREATE_CHILD_SA exchange, and the Diffie-Hellman value (if KE payloads are included in the CREATE_CHILD_SA exchange).

If a CREATE_CHILD_SA exchange includes a KEi payload, at least one of the SA offers MUST include the Diffie-Hellman group of the KEi MUST be an element of the group the initiator expects the responder to accept (additional Diffie-Hellman groups can be proposed). If the responder rejects the Diffie-Hellman group of the KEi payload, the responder MUST reject the request and indicate its preferred Diffie-Hellman group in the INVALID_KE_PAYLOAD Notification payload. In the case of such a rejection, the CREATE_CHILD_SA exchange fails, and the initiator SHOULD retry the exchange with a Diffie-Hellman proposal and KEi in the group that the responder gave in the INVALID_KE_PAYLOAD.

NEW-1.3.1 Creating New CHILD_SAs with the CREATE_CHILD_SA Exchange

A CHILD_SA may be created by sending a CREATE_CHILD_SA request. The CREATE_CHILD_SA request for creating a new CHILD_SA is:

Initiator                               Responder
-----------                               -----------
HDR, SK {SA, Ni, [KEi],
         TSi, TSr}        -->

The initiator sends SA offer(s) in the SA payload, a nonce in the NI payload, optionally a Diffie-Hellman value in the KEi payload, and the proposed traffic selectors for the proposed CHILD_SA in the TSi and TSr payloads.

The CREATE_CHILD_SA response for creating a new CHILD_SA is:

<--  HDR, SK {SA, Nr, [KEr],
         TSi, TSr}

The responder replies (using the same Message ID to respond) with the accepted offer in an SA payload, and a Diffie-Hellman value in the KEr payload if KEi was included in the request and the selected cryptographic suite includes that group.

The traffic selectors for traffic to be sent on that SA are specified in the TS payloads in the response, which may be a subset of what the initiator of the CHILD_SA proposed.

NEW-1.3.2 Rekeying IKE_SAs with the CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA request for rekeying an IKE_SA is:
The initiator sends SA offer(s) in the SA payload, a nonce in the Ni payload, and optionally a Diffie-Hellman value in the KEi payload. New initiator and responder SPIs are supplied in the SPI fields.

The CREATE_CHILD_SA response for rekeying an IKE_SA is:

\[ \text{Initiator} \rightarrow \text{Responder} \]

\[ \text{HDR, SK \{SA, Ni, [KEi]\}} \]

The responder replies (using the same Message ID to respond) with the accepted offer in an SA payload, and a Diffie-Hellman value in the KEr payload if KEi was included in the request and the selected cryptographic suite includes that group.

The new IKE_SA has its message counters set to 0, regardless of what they were in the earlier IKE_SA. The window size starts at 1 for any new IKE_SA.

NEW-1.3.3 Rekeying CHILD_SAs with the CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA request for rekeying a CHILD_SA is:

\[ \text{Initiator} \rightarrow \text{Responder} \]

\[ \text{HDR, SK \{N, SA, Ni, [KEi], TSi, TSr\}} \]

The initiator sends SA offer(s) in the SA payload, a nonce in the Ni payload, optionally a Diffie-Hellman value in the KEi payload, and the proposed traffic selectors for the proposed CHILD_SA in the TSi and TSr payloads. When rekeying an existing CHILD_SA, the leading N payload of type REKEY_SA MUST be included and MUST identify the sender’s inbound SA being rekeyed.

The CREATE_CHILD_SA response for rekeying a CHILD_SA is:

\[ \text{Initiator} \rightarrow \text{Responder} \]

\[ \text{HDR, SK \{N, SA, Ni, [KEi], TSi, TSr\}} \]

The responder replies (using the same Message ID to respond) with the accepted offer in an SA payload, and a Diffie-Hellman value in the KEr payload if KEi was included in the request and the selected cryptographic suite includes that group.
The traffic selectors for traffic to be sent on that SA are specified in the TS payloads in the response, which may be a subset of what the initiator of the CHILD_SA proposed.

3.2 Rekeying the IKE_SA vs. reauthentication

Rekeying the IKE_SA and reauthentication are different concepts in IKEv2. Rekeying the IKE_SA establishes new keys for the IKE_SA and resets the Message ID counters, but it does not authenticate the parties again (no AUTH or EAP payloads are involved).

While rekeying the IKE_SA may be important in some environments, reauthentication (the verification that the parties still have access to the long-term credentials) is often more important.

IKEv2 does not have any special support for reauthentication. Reauthentication is done by creating a new IKE_SA from scratch (using IKE_SA_INIT/IKE_AUTH exchanges, without any REKEY_SA notify payloads), creating new CHILD_SAs within the new IKE_SA (without REKEY_SA notify payloads), and finally deleting the old IKE_SA (which deletes the old CHILD_SAs as well).

This means that reauthentication also establishes new keys for the IKE_SA and CHILD_SAs. Therefore, while rekeying can be performed more often than reauthentication, the situation where "authentication lifetime" is shorter than "key lifetime" does not make sense.

While creation of a new IKE_SA can be initiated by either party (initiator or responder in the original IKE_SA), the use of EAP authentication and/or configuration payloads means in practice that reauthentication has to be initiated by the same party as the original IKE_SA. IKEv2 does not currently allow the responder to request reauthentication in this case; however, there is ongoing work to add this functionality [ReAuth].

(References: "Reauthentication in IKEv2" thread, Oct/Nov 2004.)

3.3 SPIs when rekeying the IKE_SA

Section 2.18 says that "New initiator and responder SPIs are supplied in the SPI fields”. This refers to the SPI fields in the Proposal structures inside the Security Association (SA) payloads, not the SPI fields in the IKE header.

3.4 SPI when rekeying a CHILD_SA

Section 3.10.1 says that in REKEY_SA notifications, "The SPI field identifies the SA being rekeyed."

Since CHILD_SAs always exist in pairs, there are two different SPIs. The SPI placed in the REKEY_SA notification is the SPI the exchange initiator would expect in inbound ESP or AH packets (just as in Delete payloads).

3.5 Deleting SAs

It is not clear that SAs must be actively deleted. The text sometimes says that SAs are "closed" when it means that the SAs are actively deleted. Section 1.4 says "ESP and AH SAs always exist in pairs, with one SA in each direction. When an SA is closed, both members of the pair MUST be closed." It is important to note that SAs that are closed need to be actively deleted with DELETE payloads.

4. Traffic selectors

4.1 Semantics of complex traffic selector payloads

As described in Section 3.13, the TSi/TSr payloads can include one or more individual traffic selectors.

There is no requirement that TSi and TSr contain the same number of individual traffic selectors. Thus, they are interpreted as follows: a packet matches a given TSi/TSr if it matches at least one of the individual selectors in TSi, and at least one of the individual selectors in TSr.

For instance, the following traffic selectors:

\[
\text{TSi} = ((17, 100, 192.0.1.66-192.0.1.66), \\
(17, 200, 192.0.1.66-192.0.1.66))
\]
\[
\text{TSr} = ((17, 300, 0.0.0.0-255.255.255.255), \\
(17, 400, 0.0.0.0-255.255.255.255))
\]

would match UDP packets from 192.0.1.66 to anywhere, with any of the four combinations of source/destination ports \((100,300), (100,400), (200,300), \text{and} (200,400)\).

This implies that some types of policies may require several CHILD_SA pairs. For instance, a policy matching only source/destination ports \((100,300)\) and \((200,400)\), but not the other two combinations, cannot be negotiated as a single CHILD_SA pair using IKEv2.
4.2 ICMP type/code in traffic selector payloads

The traffic selector types 7 and 8 can also refer to ICMP type and code fields. As described in Section 3.13.1, "For the ICMP protocol, the two one octet fields Type and Code are treated as a single 16 bit integer (with Type in the most significant eight bits and Code in the least significant eight bits) port number for the purposes of filtering based on this field."

This encoding is quite clear. However, as both TSi and TSr are always present, together they have two "start port" fields (one in TSi and one in TSr) and two "end port" fields. Since ICMP messages only have a single type/code field (instead of separate source/destination ports, like TCP and UDP), there is some room for confusion.

One sensible interpretation would be that in case of ICMP, the "start port" fields in TSi and TSr must always be equal, and likewise for the "end port" fields.

4.3 Mobility header in traffic selector payloads

Traffic selectors can use IP Protocol ID 135 to match the IPv6 mobility header [MIPv6]. However, the IKEv2 specification does not define how to represent the "MH Type" field in traffic selectors.

At some point, it was expected that this will be defined in a separate document later. However, [RFC2401bis] says that "For IKE, the IPv6 mobility header message type (MH type) is placed in the most significant eight bits of the 16 bit local "port" selector."

(References: Tero Kivinen’s mail "Issue #86: Add IPv6 mobility header message type as selector", 2003-10-14.)

4.4 Narrowing the traffic selectors

Section 2.9 describes how traffic selectors are negotiated when creating a CHILD_SA. A more concise summary of the narrowing process is presented below.

- If the responder’s policy does not allow any part of the traffic covered by TSi/TSr, it responds with TS_UNACCEPTABLE.

- If the responder’s policy allows the entire set of traffic covered by TSi/TSr, no narrowing is necessary, and the responder can return the same TSi/TSr values.
Otherwise, narrowing is needed. If the responder’s policy allows all traffic covered by TSi[1]/TSr[1] (the first traffic selectors in TSi/TSr) but not entire TSi/TSr, the responder narrows to an acceptable subset of TSi/TSr that includes TSi[1]/TSr[1].

If the responder’s policy does not allow all traffic covered by TSi[1]/TSr[1], but does allow some parts of TSi/TSr, it narrows to an acceptable subset of TSi/TSr.

In the last two cases, there may be several subsets that are acceptable (but their union is not); in this case, the responder arbitrarily chooses one of them, and includes ADDITIONAL_TS_POSSIBLE notification in the response.

4.5 SINGLE_PAIR_REQUIRED

The description of the SINGLE_PAIR_REQUIRED notify payload in Sections 2.9 and 3.10.1 is not fully consistent.

We do not attempt to describe this payload in this document either, since it is expected that most implementations will not have policies that require separate SAs for each address pair.

Thus, if only some part (or parts) of the TSi/TSr proposed by the initiator is (are) acceptable to the responder, most responders should simply narrow TSi/TSr to an acceptable subset (as described in the last two paragraphs of Section 2.9), rather than use SINGLE_PAIR_REQUIRED.

4.6 Traffic selectors violating own policy

Section 2.9 describes traffic selector negotiation in great detail. One aspect of this negotiation that may need some clarification is that when creating a new SA, the initiator should not propose traffic selectors that violate its own policy. If this rule is not followed, valid traffic may be dropped.

This is best illustrated by an example. Suppose that host A has a policy whose effect is that traffic to 192.0.1.66 is sent via host B encrypted using AES, and traffic to all other hosts in 192.0.1.0/24 is also sent via B, but must use 3DES. Suppose also that host B accepts any combination of AES and 3DES.

If host A now proposes an SA that uses 3DES, and includes TSr containing (192.0.1.0-192.0.1.0.255), this will be accepted by host B. Now, host B can also use this SA to send traffic from 192.0.1.66, but those packets will be dropped by A since it requires the use of AES for those traffic. Even if host A creates a new SA only for
192.0.1.66 that uses AES, host B may freely continue to use the first SA for the traffic. In this situation, when proposing the SA, host A should have followed its own policy, and included a TSr containing ((192.0.1.0-192.0.1.65),(192.0.1.67-192.0.1.255)) instead.

In general, if (1) the initiator makes a proposal "for traffic X (TSi/TSr), do SA", and (2) for some subset X' of X, the initiator does not actually accept traffic X' with SA, and (3) the initiator would be willing to accept traffic X' with some SA' (!=SAi), valid traffic can be unnecessarily dropped since the responder can apply either SA or SA' to traffic X'.


5. Configuration payloads

5.1 Length of configuration attribute type field

In Section 3.15.1, Figure 23 shows that the length of the "Attribute Type" field is 15 bits, while the text below the figure says the length is 7 bits.

The figure is correct, the field is 15 bits.

(References: Tero Kivinen’s mail "Comments to the draft-ietf-ipsec-ikev2-11.txt", 2003-11-09. Yoav Nir’s mail "Will ikev2-16 be the charm?", 2004-09-23. Charlie Kaufman’s mail"draft-ietf-ipsec-ikev2-17.txt", 2004-10-04. It is expected that this issue will be fixed during the "Authors’ 48 hours" before the RFC is published.)

5.2 Requesting any INTERNAL_IP4/IP6_ADDRESS

When describing the INTERNAL_IP4/IP6_ADDRESS attributes, Section 3.15.1 says that "In a request message, the address specified is a requested address (or zero if no specific address is requested)". The question here is that does "zero" mean an address "0.0.0.0" or a zero length string?

Earlier, the same section also says that "If an attribute in the CFG_REQUEST Configuration Payload is not zero length it is taken as a suggestion for that attribute". Also, the table of configuration attributes shows that the length of INTERNAL_IP4_ADDRESS is either "0 or 4 octets", and likewise, INTERNAL_IP6_ADDRESS is either "0 or 17 octets".
Thus, if the client does not request a specific address, it includes a zero-length INTERNAL_IP4/IP6_ADDRESS attribute, not an attribute containing an all-zeroes address. The example in 2.19 is thus incorrect, since it shows the attribute as "INTERNAL_ADDRESS(0.0.0.0)".

However, since the value is only a suggestion, implementations are recommended to ignore suggestions they do not accept; or in other words, treat the same way a zero-length INTERNAL_IP4_ADDRESS, "0.0.0.0", and any other addresses the implementation does not recognize as a reasonable suggestion.

5.3 INTERNAL_IP4_SUBNET/INTERNAL_IP6_SUBNET

Section 3.15.1 describes the INTERNAL_IP4_SUBNET as "The protected sub-networks that this edge-device protects. This attribute is made up of two fields; the first being an IP address and the second being a netmask. Multiple sub-networks MAY be requested. The responder MAY respond with zero or more sub-network attributes." INTERNAL_IP6_SUBNET is defined in a similar manner.

This raises two questions: first, since this information is usually included in the TSr payload, what functionality does this attribute add? And second, what does this attribute mean in CFG_REQUESTs?

For the first question, there seem to be two sensible interpretations. Clearly TSr (in IKE_AUTH or CREATE_CHILD_SA response) indicates which subnets are accessible through the SA that was just created.

The first interpretation of the INTERNAL_IP4/6_SUBNET attributes is that they indicate additional subnets that can be reached through this gateway, but need a separate SA. According to this interpretation, the INTERNAL_IP4/6_SUBNET attributes are useful mainly when they contain addresses not included in TSr.

The second interpretation is that the INTERNAL_IP4/6_SUBNET attributes express the gateway’s policy about what traffic should be sent through the gateway. The client can choose whether other traffic (covered by TSr, but not in INTERNAL_IP4/6_SUBNET) is sent through the gateway or directly the destination. According to this interpretation, the attributes are useful mainly when TSr contains addresses not included in the INTERNAL_IP4/6_SUBNET attributes.

These two interpretations are not totally incompatible: in both cases, they suggest that traffic to the addresses listed in the INTERNAL_IP4/6_SUBNET attributes should be sent via this gateway (and, of course, the packets have to be sent over some SA whose
traffic selectors cover the address in question).

A couple of examples are given below. For instance, if there are two subnets, 192.0.1.0/26 and 192.0.2.0/24, and the client’s request contains the following:

\[
\text{CP(CFG REQUEST)} = \\
\text{INTERNAL IP4 ADDRESS()} \\
\text{TSi = (0, 0-65536, 0.0.0.0-255.255.255.255)} \\
\text{TSr = (0, 0-65536, 0.0.0.0-255.255.255.255)}
\]

Then a valid response could be the following (in which TSr and INTERNAL_IP4_SUBNET contain the same information):

\[
\text{CP(CFG REPLY)} = \\
\text{INTERNAL IP4 ADDRESS(192.0.1.234)} \\
\text{INTERNAL IP4 SUBNET(192.0.1.0/255.255.255.192)} \\
\text{INTERNAL IP4 SUBNET(192.0.2.0/255.255.255.0)} \\
\text{TSi = (0, 0-65536, 192.0.1.234-192.0.1.234)} \\
\text{TSr = ((0, 0-65536, 192.0.1.0-192.0.1.63),} \\
\text{(0, 0-65536, 192.0.2.0-192.0.2.255))}
\]

In these cases, the INTERNAL_IP4_SUBNET does not really carry any useful information. Another possible reply would have been this:

\[
\text{CP(CFG REPLY)} = \\
\text{INTERNAL IP4 ADDRESS(192.0.1.234)} \\
\text{INTERNAL IP4 SUBNET(192.0.1.0/255.255.255.192)} \\
\text{INTERNAL IP4 SUBNET(192.0.2.0/255.255.255.0)} \\
\text{TSi = (0, 0-65536, 192.0.1.234-192.0.1.234)} \\
\text{TSr = (0, 0-65536, 0.0.0.0-255.255.255.255)}
\]

This would mean that the client can send all its traffic through the gateway, but the gateway does not mind if the client sends traffic not included by INTERNAL_IP4_SUBNET directly to the destination (without going through the gateway).

A different situation arises if the gateway has a policy that requires the traffic for the two subnets to be carried in separate SAs. Then a response like this would indicate to the client that if it wants access to the second subnet, it needs to create a separate SA:

\[
\text{CP(CFG REPLY)} = \\
\text{INTERNAL IP4 ADDRESS(192.0.1.234)} \\
\text{INTERNAL IP4 SUBNET(192.0.1.0/255.255.255.192)} \\
\text{INTERNAL IP4 SUBNET(192.0.2.0/255.255.255.0)} \\
\text{TSi = (0, 0-65536, 192.0.1.234-192.0.1.234)}
\]
TSr = (0, 0-65536, 192.0.1.0-192.0.1.63)

INTERNAL_IP4_SUBNET can also be useful if the client’s TSr included only part of the address space. For instance, if the client requests the following:

\[
\begin{align*}
\text{CP(CFG\_REQUEST)} &= \text{INTERNAL\_IP4\_ADDRESS}() \\
\text{TSi} &= (0, 0-65536, 0.0.0.0-255.255.255.255) \\
\text{TSr} &= (0, 0-65536, 192.0.2.155-192.0.2.155)
\end{align*}
\]

Then the gateway’s reply could be this:

\[
\begin{align*}
\text{CP(CFG\_REPLY)} &= \text{INTERNAL\_IP4\_ADDRESS}(192.0.1.234) \\
& \quad \text{INTERNAL\_IP4\_SUBNET}(192.0.1.0/255.255.255.192) \\
& \quad \text{INTERNAL\_IP4\_SUBNET}(192.0.2.0/255.255.255.0) \\
\text{TSi} &= (0, 0-65536, 192.0.1.234-192.0.1.234) \\
\text{TSr} &= (0, 0-65536, 192.0.2.155-192.0.2.155)
\end{align*}
\]

It is less clear what the attributes mean in CFG\_REQUESTs, and whether other lengths than zero make sense in this situation (but for INTERNAL\_IP6\_SUBNET, zero length is not allowed at all!). Currently this document recommends that implementations should not include INTERNAL\_IP4\_SUBNET or INTERNAL\_IP6\_SUBNET attributes in CFG\_REQUESTs.

For the IPv4 case, this document recommends using only netmasks consisting of some amount of "1" bits followed by "0" bits; for instance, "255.0.255.0" would not be a valid netmask for INTERNAL\_IP4\_SUBNET.


5.4 INTERNAL\_IP4\_NETMASK

Section 3.15.1 defines the INTERNAL\_IP4\_NETMASK attribute, and says that "The internal network’s netmask. Only one netmask is allowed in the request and reply messages (e.g., 255.255.255.0) and it MUST be used only with an INTERNAL\_IP4\_ADDRESS attribute".

However, it is not clear what exactly this attribute means, as the concept of "netmask" is not very well defined for point-to-point links (unlike multi-access links, where it means "you can reach hosts..."
inside this netmask directly using layer 2, instead of sending packets via a router").

One possible interpretation would be that the host is given a whole block of IP addresses instead of a single address. This is also what Framed-IP-Netmask does in [RADIUS] and the IPCP "subnet mask" extension does in PPP [IPCPSubnet]. This interpretation would also work nicely with IPv6 (see the following section).

However, one IKEv2 guru assured the authors that this interpretation is not correct. Section 3.15.1 also says that multiple addresses are assigned using multiple INTERNAL_IP4_ADDRESS attributes.

Currently, this document’s interpretation is the following:

- INTERNAL_IP4_NETMASK in a CFG_REPLY means exactly the same thing as INTERNAL_IP4_SUBNET containing the same information (see the previous section for description of INTERNAL_IP4_SUBNET).

- INTERNAL_IP4_NETMASK does not make sense for CFG_REQUESTs, and the example in Section 2.19 is incorrect in this sense. (Another interpretation would be that by sending, for instance, the combination of INTERNAL_IP4_ADDRESS(192.0.2.0) and INTERNAL_IP4_NETMASK(255.255.255.0), the client is asking to be assigned one IP address from the network 192.0.2.0/24. However, this interpretation is not supported by the IKEv2 spec.)

This interpretation is not yet settled; and it would imply that the whole attribute is totally unnecessary.

Yet another possible interpretation would be that INTERNAL_IP4_NETMASK indicates a broadcast address, meaning that if a client sends a packet to this address, the gateway will decrypt it and send copies to all other VPN clients in that address range. However, no implementation is known to do this, and there is nothing in the IKEv2 spec that would support this interpretation.

Fortunately, Section 4 clearly says that a minimal implementation does not need to include or understand the INTERNAL_IP4_NETMASK attribute, and thus this document recommends that implementations should not use the INTERNAL_IP4_NETMASK attribute at all.

5.5 Configuration payloads for IPv6

IKEv2 also defines configuration payloads for IPv6. However, they are based on the corresponding IPv4 payloads, and do not fully follow the "normal IPv6 way of doing things".

A client can be assigned an IPv6 address using the INTERNAL_IP6_ADDRESS configuration payload. Presumably, the idea was that a minimal exchange would look something like this:

```
CP(CFG_REQUEST) =
    INTERNAL_IP6_ADDRESS()
    INTERNAL_IP6_DNS()
TSi = (0, 0-65536, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)
TSr = (0, 0-65536, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)

CP(CFG_REPLY) =
    INTERNAL_IP6_ADDRESS(2001:DB8:0:1:2:3:4:5/?)
TSi = (0, 0-65536, 2001:DB8:0:1:2:3:4:5 - 2001:DB8:0:1:2:3:4:5)
TSr = (0, 0-65536, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)
```

In particular, IPv6 stateless autoconfiguration or router advertisement messages are not used; neither is neighbor discovery.

While this approach is reasonably simple, it has some limitations: IPsec tunnels configured using IKEv2 are not fully-featured "interfaces" in the IPv6 addressing architecture [IPv6Addr] sense. In particular, they do not necessarily have link-local addresses, and this may complicate the use of protocols that assume them, such as [MLDv2]. (Whether they are called "interfaces" in some particular operating system is a different issue.)

(References: "VPN remote host configuration IPv6 ?" thread, May 2004.)

5.6 INTERNAL_IP6_ADDRESS prefix length

Earlier versions of the IKEv2 draft had an INTERNAL_IP6_NETMASK attribute corresponding to INTERNAL_IP4_NETMASK, but this was deleted when the prefix length field was added to the INTERNAL_IP6_ADDRESS attribute. Thus, it seems logical to assume that their purpose would be similar; however, this is far from obvious.

The draft quite clearly says that the client is assigned an IPv6 address using the INTERNAL_IP6_ADDRESS attribute. However, as with the netmask in IPv4, it is not clear what the prefix length here means.
Again, one possible interpretation is that a prefix length smaller than 128 in a CFG_REPLY means that the client is assigned a whole block of IPv6 addresses. This would be in line with the IPv6 addressing architecture in general, and with, e.g., the Framed-IPv6-Prefix attribute in [RADIUS6]. However, the previous section rejected this interpretation for IPv4, so it would seem strange to adopt it only for IPv6.

Thus, if we assume that INTERNAL_IP4_NETMASK and the prefix length in INTERNAL_IP6_ADDRESS have the same meaning, and the reasoning in the previous section is correct, then a CFG_REPLY containing a prefix length smaller than 128 has the same purpose as INTERNAL_IP6_SUBNET.

However, CFG_REQUESTs are more complicated. It seems that a CFG_REQUEST message that requests a specific IPv6 address (usually an address this client was using earlier) should have prefix length 128. But what do other prefix lengths mean in CFG_REQUESTs?

Section 3.15.1 says that "With IPv6, a requestor MAY supply the low order address bytes it wants to use": presumably the prefix length tells how many low order bits there are (i.e., if the prefix length is X, the requester supplies 128-X low order address bits). However, this is quite confusing: if, say, a prefix length 126 means that "I want to use these 128-126=2 low order bits", why does prefix length 128 mean that "I want to use these 128 low order bits"?

Another interpretation is that instead of "low order", the draft should have said "high order", and thus a prefix length smaller than 128 means "I’d like to get an address from this subnet".

Given this very confusing discussion, this document recommends that implementations should not use other INTERNAL_IP6_ADDRESS prefix lengths than 128.

5.7 INTERNAL_IP6_NBNS

Section 3.15.1 defines the INTERNAL_IP6_NBNS attribute for sending the IPv6 address of NetBIOS name servers.

However, NetBIOS is not defined for IPv6, and probably never will be. Thus, this attribute most likely does not make much sense.

(Pointed out by Bernard Aboba in the IP Configuration Security (ICOS) BoF at IETF62.)

6. Miscellaneous issues
6.1 Diffie-Hellman for first CHILD_SA

Section 1.2 shows that IKE_AUTH messages do not contain KEi/KEr or Ni/Nr payloads. This implies that the SA payload in IKE_AUTH exchange cannot contain Transform Type 4 (Diffie-Hellman Group) with any other value than NONE.

6.2 Extended Sequence Numbers (ESN) transform

The description of the ESN transform in Section 3.3 has been proved difficult to understand. When the ESN transform is included, it has the following meaning:

- A proposal containing one ESN transform with value 0 means "do not use extended sequence numbers".
- A proposal containing one ESN transform with value 1 means "use extended sequence numbers".
- A proposal containing two ESN transforms with values 0 and 1 means "I support both normal and extended sequence numbers, you choose". (Obviously this case is only allowed in requests; the response will contain only one ESN transform.)

In most cases, the exchange initiator will include either the first or third alternative in its SA payload. The second alternative is rarely useful for the initiator: it means that using normal sequence numbers is not acceptable (so if the responder does not support ESNs, the exchange will fail with NO_PROPOSAL_CHOSEN).

Section 3.3.2 also says that "If Transform Type 5 is not included in a proposal, use of Extended Sequence Numbers is assumed". Or in other words, omitting the ESN transform means the same thing as including one ESN transform with value 1.

This choice of default value is somewhat counterintuitive and as described above, rarely useful. There is ongoing discussion about making including the ESN transform mandatory, thus removing this illogical default value.

(References: "Technical change needed to IKEv2 before publication" and "STRAW POLL: Dealing with the ESN negotiation interop issue in IKEv2" threads, March 2005.)

6.3 Matching ID_IPV4_ADDR and ID_IPV6_ADDR

When using the ID_IPV4_ADDR/ID_IPV6_ADDR identity types in IDi/IDr payloads, IKEv2 does not require this address to match the address in
the IP header (of IKEv2 packets), or anything in the TSi/TSr payloads. The contents of IDi/IDr is used purely to fetch the policy and authentication data related to the other party.

(References: "Identities types IP address, FQDN/user FQDN and DN and its usage in preshared key authentication" thread, Jan 2005.)

6.4 Relationship of IKEv2 to RFC2401bis

The IKEv2 document refers to [RFC2401bis], but it never makes clear what the exact relationship is. That is probably because there is no exact relationship. However, the IKEv2 document could state this explicitly.

IKEv2 can be used with either [RFC2401] or [RFC2401bis], except in places that are only covered by [RFC2401bis]. The areas specific to [RFC2401bis] are ECN (Section 2.24), fragmentation control (Section 3.10), and the use of opaque ports in traffic selectors (Section 3.13.1). IKEv2 allows the creation of a single SA that has multiple protocols when being used with [RFC2401]; this is not allowed in [RFC2401bis].

6.5 Reducing the window size

In IKEv2, the window size is assumed to be a (possibly configurable) property of a particular implementation, and is not related to congestion control (unlike the window size in TCP, for instance).

In particular, there is no way to reduce the window size of an existing IKE_SA. However, when rekeying an IKE_SA, the new IKE_SA starts with window size 1 until it is explicitly increased by sending a new SET_WINDOW_SIZE notification.

6.6 Minimum size of nonces

Section 2.10 says that "Nonces used in IKEv2 MUST be randomly chosen, MUST be at least 128 bits in size, and MUST be at least half the key size of the negotiated prf."

However, the initiator chooses the nonce before the outcome of the negotiation is known. In this case, the nonce has to be long enough for all the PRFs being proposed.

6.7 Initial zero octets on port 4500

It is not clear whether a peer sending an IKE_SA_INIT request on port 4500 should include the initial four zero octets. Section 2.23 talks about how to upgrade to tunneling over port 4500 after message 2, but
The very beginning of Section 2 says "... though IKE messages may also be received on UDP port 4500 with a slightly different format (see section 2.23)."

That "slightly different format" is only described in discussing what to do after changing to port 4500. However, [RFC3948] shows clearly the format has the initial zeros even for initiators on port 4500. Furthermore, without the initial zeros, the processing engine cannot determine whether the packet is an IKE packet or an ESP packet.

Thus, all packets sent on port 4500 need the four zero prefix; otherwise, the receiver won't know how to handle them.

**6.8 SPI values in IKE_SA_INIT exchange**

Normal IKE messages include the initiator’s and responder’s SPIs, both of which are non-zero, in the IKE header. However, there are some corner cases where the IKEv2 specification is not fully consistent about what values should be used.

First, Section 3.1 says that the Responder’s SPI "...MUST NOT be zero in any other message" (than the first message of the IKE_SA_INIT exchange). However, the figure in Section 2.6 shows the second IKE_SA_INIT message as "HDR(A,0), N(COOKIE)", contradicting the text in 3.1.

Since the responder’s SPI identifies security-related state held by the responder, and in this case no state is created, sending a zero value seems reasonable.

Second, in addition to cookies, there are several other cases when the IKE_SA_INIT exchange does not result in the creation of an IKE_SA (for instance, INVALID_KE_PAYLOAD or NO_PROPOSAL_CHOSEN). What
responder SPI value should be used in the IKE_SA_INIT response in this case?

Since the IKE_SA_INIT request always has a zero responder SPI, the value will not be actually used by the initiator. Thus, we think sending a zero value is correct also in this case.

6.9 SPI values for messages outside of an IKE_SA

The IKEv2 specification does not say what SPI values should be used for Informational requests sent outside of an IKE_SA.

There seem to be two cases when such a message can be sent: INVALID_IKE_SPI and INVALID_SPI notifications. Especially in the latter case, no meaningful IKE SPI values are available.

A strict interpretation of the specification would require the sender to invent garbage values for the SPI fields. However, we think this was not the intention, and using zero values is acceptable.

6.10 Protocol ID/SPI fields in Notify payloads

Section 3.10 says that the Protocol ID field in Notify payloads "For notifications which do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt". However, the specification does not clearly say which notifications are related to existing SAs and which are not.

Since the main purpose of the Protocol ID field is to specify the type of the SPI, our interpretation is that the Protocol ID field should be non-zero only when the SPI field is non-empty.

There are currently only two notifications where this is the case: INVALID_SELECTORS and REKEY_SA.

6.11 INVALID_IKE_SPI

Section 3.10.1 says that the INVALID_IKE_SPI notification "indicates an IKE message was received with an unrecognized destination SPI. This usually indicates that the recipient has rebooted and forgotten the existence of an IKE_SA."

The text does not say whether the SPI value should be included in the notification. However, it is clear that the notification will be useful to the recipient only if it can find the IKE_SA somehow, so the SPI should be included.

This still leaves two questions open: which SPI(s) should be
included, and how it (or they) should be sent. For the first
tquestion, the alternatives are the unrecognized destination SPI, the
source SPI (which presumably would be more useful for the recipient),
or both. For the second question, the SPI(s) could be placed in the
SPI field(s) in the IKE header, the SPI field in the Notify payload,
or the Notification Data field.

In the case of another related notification, INVALID_SPI, the
situation is clearer: there is only a single SPI, and the text
explicitly says that the SPI is sent as Notification Data (since the
notification is not about an existing SA, the SPI field in the Notify
payload is not used; and obviously the value cannot be placed in the
IKE header).

Since the INVALID_IKE_SPI notification is sent outside of an IKE_SA,
and it is not about an existing SA, it seems that using Notification
Data would be the logical choice. However, this issue needs more
discussion and we do not yet propose any solution in this document.

6.12 Which message should contain INITIAL_CONTACT

The description of the INITIAL_CONTACT notification in Section 3.10.1
says that "This notification asserts that this IKE_SA is the only
IKE_SA currently active between the authenticated identities". However,
neither Section 2.4 nor 3.10.1 says in which message this
payload should be placed.

The text does talk about authenticated identities, so it seems the
notification cannot be sent before both endpoints have been
authenticated. Thus, the possible places are the last IKE_AUTH
response message and a separate Informational exchange.

Based on how this was implemented in IKEv1, it seems the intent was
to use a separate Informational exchange.

7. Status of the clarifications

This document is work-in-progress, and it contains both relatively
stable and finished parts, and other parts that are incomplete or
even incorrect. To help the reader in deciding how much weight
should be given to each clarification, this section contains our
opinions about which parts we believe to are stable, and which are
likely to change in future versions.

Those clarifications believed to be correct and without controversy
are marked with three asterisks (***); those where the clarification
is known to be incomplete and/or there is disagreement about what the
correct interpretation is are marked with one asterisk (*). The
clarifications marked with two asterisks (**) are somewhere between the extremes.

2. Authentication
   2.1 Data included in AUTH payload calculation ***
   2.2 Hash function for RSA signatures ***
   2.3 Encoding method for RSA signatures ***
   2.4 Identification type for EAP ***
   2.5 Identity for policy lookups when using EAP ***
   2.6 EAP authentication and the AUTH payload ***
   2.7 Certificate encoding types ***
   2.8 Shared key authentication and fixed PRF key size **
   2.9 EAP authentication and fixed PRF key size *

3. Keying and rekeying
   3.1 Semantics of the CREATE_CHILD_SA exchange *
   3.2 Rekeying the IKE_SA vs. reauthentication **
   3.3 SPIs when rekeying the IKE_SA ***
   3.4 Which SPI to use in REKEY_SA ***
   3.5 Deleting SAs **

4. Traffic selectors
   4.1 Semantics of complex traffic selector payloads ***
   4.2 ICMP type/code in traffic selector payloads ***
   4.3 Mobility header in traffic selector payloads ***
   4.4 Narrowing the traffic selectors ***
   4.5 SINGLE_PAIR_REQUIRED **
   4.6 Traffic selectors violating own policy *

5. Configuration payloads
   5.1 Length of configuration attribute type field ***
   5.2 Requesting any INTERNAL_IP4/IP6_ADDRESS ***
   5.3 INTERNAL_IP4_SUBNET/INTERNAL_IP6_SUBNET **
   5.4 INTERNAL_IP4_NETMASK **
   5.5 Configuration payloads for IPv6 *
   5.6 INTERNAL_IP6_ADDRESS prefix length *
   5.7 INTERNAL_IP6_NBNS ***

6. Miscellaneous issues
   6.1 Diffie-Hellman for first CHILD_SA ***
   6.2 Extended Sequence Numbers (ESN) transform **
   6.3 Matching ID_IPV4_ADDR and ID_IPV6_ADDR ***
   6.4 Relationship of IKEv2 to RFC2401bis **
   6.5 Reducing the window size **
   6.6 Minimum size of nonces ***
   6.7 Initial zero octets on port 4500 ***
   6.8 SPI values in IKE_SA_INIT exchange **
   6.9 SPI values for messages outside of an IKE_SA *
   6.10 Protocol ID/SPI fields in Notify payloads **
   6.11 INVALID_IKE_SPI *
   6.12 Which message should contain INITIAL_CONTACT **
Future versions of this document will, of course, change these estimates (and changes in both directions are possible, though hopefully it’s more towards higher confidence).

8. Implementation mistakes

Some implementers at the first IKEv2 bakeoff didn’t do everything correctly. This may seem like an obvious statement, but it is probably useful to list a few things that were clear in the document and not needing clarification, that some implementors didn’t do. All of these things caused interoperability problems.

- Some implementations continued to send traffic on a CHILD_SA after it was rekeyed, even after receiving an DELETE payload.

- After rekeying an IKE_SA, some implementations did not reset their message counters to zero. One set the counter to 2, another did not reset the counter at all.

- Some implementations could only handle a single pair of traffic selectors, or would only process the first pair in the proposal.

9. Open issues

This section lists issues that this document probably should address, but has not done so yet.

- In the traffic selector discussion, we need to come up with better wording for the "sender’s inbound" SAs. Is that traffic that is being sent to the sender, or traffic being sent from the sender?

- Many of the configuration payload issues in this draft are still far from clear. These need to be resolved before implementers can feel assured of creating interoperable implementations.

- There may need to be more text about deleting an old SA after rekeying is finished.

- The text about sending a DELETE for only one direction of an SA (and the responder sending the DELETE for the other direction of the SA in its response) doesn’t explain the logic, and doesn’t say why you should not send DELETEs for both directions of the SA. We need to add a description of the race condition if one side deletes both SAs at once.

- When the document uses the term "original initiator" (or similar terms), it is not clear whether or not that term also applies to
the side that originates an rekeying of the IKE_SA. That is, if
Alice starts the first IKE_SA, but then Bob rekeys the IKE_SA, is
the "original initiator" from that point on now Bob, or is it
still Alice? Also, if it is Bob, at what exact point does it
become Bob? This needs to be cleared up on a case-by-case basis
throughout the document.

- It would be very useful to have actual examples of certificate
type 12 (hash and URL of X.509 certificates) and type 13 (hash and
URL of X.509 bundle).

- The message IDs of IKE_SA_INIT messages is unclear if there are
cookies followed by INVALID_KE_PAYLOAD. See "Question about
N(COOKIE) and N(INVALID_KE_PAYLOAD) combination" thread, Oct 2004.
This definitely needs to be clarified.

- There is some confusion on when to emit and process
INITIAL_CONTACT payloads. References: Michael Richardson’s mail
"Initial Contact Messages", 2004-12-05. Paul Hoffman’s reply,
identities" thread in Jan 2005.) It is not clear if there is an
interoperability issue because reacting to INITIAL_CONTACT is
optional, but this should be cleared up.

10. Security considerations

This document does not introduce any new security considerations to
IKEv2. If anything, clarifying complex areas of the specification
can reduce the likelihood of implementation problems that may have
security implications.

11. IANA considerations

This document does not change or create any IANA-registered values.

12. Acknowledgments

This document is mainly based on conversations on the IPsec WG
mailing list. The authors would especially like to thank Bernard
Aboba, Jari Arkko, Vijay Devarapalli, William Dixon, Mika
Joutsenvirta, Charlie Kaufman, Tero Kivinen, Yoav Nir, and Michael
Richardson for their contributions.

In addition, the authors would like to thank all the participants of
the first public IKEv2 bakeoff, held in Santa Clara in February 2005,
for their questions and proposed clarifications.
13. References

13.1 Normative References


13.2 Informative References


IPv6Addr

MIPv6

MLDv2

NAI

NAIbis

RADEAP

RADIUS

RADIUS6

RFC2119
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

RFC2822

RFC3948

RFC822

ReAuth

SCVP
Freeman, T., Housley, R. and A. Malpani, "Simple Certificate Validation Protocol (SCVP)"
Authors’ Addresses

Pasi Eronen
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
Finland

EMail: pasi.eronen@nokia.com

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060
USA

EMail: paul.hoffman@vpnc.org
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.