The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX

Status of This Memo

This document specifies an Internet standards track protocol for the
Internet community, and requests discussion and suggestions for
improvements. Please refer to the current edition of the "Internet
Official Protocol Standards" (STD 1) for the standardization state
and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Internet Key Exchange (IKE) and Public Key Infrastructure for
X.509 (PKIX) certificate profile both provide frameworks that must be
profiled for use in a given application. This document provides a
profile of IKE and PKIX that defines the requirements for using PKI
technology in the context of IKE/IPsec. The document complements
protocol specifications such as IKEv1 and IKEv2, which assume the
existence of public key certificates and related keying materials,
but which do not address PKI issues explicitly. This document
addresses those issues. The intended audience is implementers of PKI
for IPsec.

Table of Contents

1. Introduction

   IKE [1], ISAKMP [2], and IKEv2 [3] provide a secure key exchange
   mechanism for use with IPsec [4] [14]. In many cases, the peers
   authenticate using digital certificates as specified in PKIX [5].
   Unfortunately, the combination of these standards leads to an
   underspecified set of requirements for the use of certificates in the
   context of IPsec.

   ISAKMP references the PKIX certificate profile but, in many cases,
   merely specifies the contents of various messages without specifying
   their syntax or semantics. Meanwhile, the PKIX certificate profile
   provides a large set of certificate mechanisms that are generally
   applicable for Internet protocols, but little specific guidance for
   IPsec. Given the numerous underspecified choices, interoperability
   is hampered if all implementers do not make similar choices, or at
   least fail to account for implementations that have chosen
   differently.

   This profile of the IKE and PKIX frameworks is intended to provide an
   agreed-upon standard for using PKI technology in the context of IPsec
   by profiling the PKIX framework for use with IKE and IPsec, and by
   documenting the contents of the relevant IKE payloads and further
   specifying their semantics.

   In addition to providing a profile of IKE and PKIX, this document
   attempts to incorporate lessons learned from recent experience with
   both implementation and deployment, as well as the current state of
   related protocols and technologies.

   Material from ISAKMP, IKEv1, IKEv2, or PKIX is not repeated here, and
   readers of this document are assumed to have read and understood
   those documents. The requirements and security aspects of those
   documents are fully relevant to this document as well.

   This document is organized as follows. Section 2 defines special
   terminology used in the rest of this document, Section 3 provides the
   profile of IKEv1/ISAKMP, Section 4 provides a profile of IKEv2, and
   Section 5 provides the profile of PKIX. Section 6 covers conventions
   for the out-of-band exchange of keying materials for configuration
   purposes.

2. Terms and Definitions

   Except for those terms that are defined immediately below, all terms
   used in this document are defined in either the PKIX [5], ISAKMP [2],
   IKEv1 [1], IKEv2 [3], or Domain of Interpretation (DOI) [6]
   documents.

o Peer source address: The source address in packets from a peer.
   This address may be different from any addresses asserted as the
   "identity" of the peer.

o FQDN: Fully qualified domain name.

o ID_USER_FQDN: IKEv2 renamed ID_USER_FQDN to ID_RFC822_ADDR. Both
   are referred to as ID_USER_FQDN in this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [7].

3. Use of Certificates in RFC 2401 and IKEv1/ISAKMP

3.1. Identification Payload

The Identification (ID) Payload indicates the identity claimed by the
sender. The recipient can then use the ID as a lookup key for policy
and for certificate lookup in whatever certificate store or directory
that it has available. Our primary concern in this section is to
profile the ID payload so that it can be safely used to generate or
lookup policy. IKE mandates the use of the ID payload in Phase 1.

The DOI [6] defines the 11 types of Identification Data that can be
used and specifies the syntax for these types. These are discussed
below in detail.

The ID payload requirements in this document cover only the portion
of the explicit policy checks that deal with the Identification
Payload specifically. For instance, in the case where ID does not
contain an IP address, checks such as verifying that the peer source
address is permitted by the relevant policy are not addressed here,
as they are out of the scope of this document.

Implementations SHOULD populate ID with identity information that is
contained within the end-entity certificate. Populating ID with
identity information from the end-entity certificate enables
recipients to use ID as a lookup key to find the peer end-entity
certificate. The only case where implementations may populate ID
with information that is not contained in the end-entity certificate
is when ID contains the peer source address (a single address, not a
subnet or range).

Because implementations may use ID as a lookup key to determine which
policy to use, all implementations MUST be especially careful to
verify the truthfulness of the contents by verifying that they
correspond to some keying material demonstrably held by the peer.

Failure to do so may result in the use of an inappropriate or
insecure policy. The following sections describe the methods for
performing this binding.

The following table summarizes the binding of the Identification
Payload to the contents of end-entity certificates and of identity
information to policy. Each ID type is covered more thoroughly in
the following sections.

ID type	Support for send	Correspond PKIX Attrib	Cert matching	SPD lookup rules
IP*_ADDR	MUST [a]	SubjAltName iPAddress	MUST [b]	[c], [d]
FQDN	MUST [a]	SubjAltName dNSName	MUST [b]	[c], [d]
USER_FQDN	MUST [a]	SubjAltName rfc822Name	MUST [b]	[c], [d]
IP range	MUST NOT	n/a	n/a	n/a
DN	MUST [a]	Entire Subject, bitwise compare	MUST [b]	MUST support lookup on any combination of C, CN, O, or OU
GN	MUST NOT	n/a	n/a	n/a
KEY_ID	MUST NOT	n/a	n/a	n/a

[a] = Implementation MUST have the configuration option to send this
      ID type in the ID payload. Whether or not the ID type is used
      is a matter of local configuration.

[b] = The ID in the ID payload MUST match the contents of the
      corresponding field (listed) in the certificate exactly, with
      no other lookup. The matched ID MAY be used for Security
      Policy Database (SPD) lookup, but is not required to be used
      for this.

[c] = At a minimum, Implementation MUST be capable of being
      configured to perform exact matching of the ID payload contents
      to an entry in the local SPD.

   [d] = In addition, the implementation MAY also be configurable to
         perform substring or wildcard matches of ID payload contents to
         entries in the local SPD. (More on this in Section 3.1.5.)

   When sending an IPV4_ADDR, IPV6_ADDR, FQDN, or USER_FQDN,
   implementations MUST be able to be configured to send the same string
   as it appears in the corresponding SubjectAltName extension. This
   document RECOMMENDS that deployers use this configuration option.
   All these ID types are treated the same: as strings that can be
   compared easily and quickly to a corresponding string in an explicit
   value in the certificate. Of these types, FQDN and USER_FQDN are
   RECOMMENDED over IP addresses (see discussion in Section 3.1.1).

   When sending a Distinguished Name (DN) as ID, implementations MUST
   send the entire DN in ID. Also, implementations MUST support at
   least the C, CN, O, and OU attributes for SPD matching. See Section
   3.1.5 for more details about DN, including SPD matching.

   Recipients MUST be able to perform SPD matching on the exact contents
   of the ID, and this SHOULD be the default setting. In addition,
   implementations MAY use substrings or wildcards in local policy
   configuration to do the SPD matching against the ID contents. In
   other words, implementations MUST be able to do exact matches of ID
   to SPD, but MAY also be configurable to do substring or wildcard
   matches of ID to SPD.

3.1.1. ID_IPV4_ADDR and ID_IPV6_ADDR

   Implementations MUST support at least the ID_IPV4_ADDR or
   ID_IPV6_ADDR ID type, depending on whether the implementation
   supports IPv4, IPv6, or both. These addresses MUST be encoded in
   "network byte order", as specified in IP [8]: The least significant
   bit (LSB) of each octet is the LSB of the corresponding byte in the
   network address. For the ID_IPV4_ADDR type, the payload MUST contain
   exactly four octets [8]. For the ID_IPV6_ADDR type, the payload MUST
   contain exactly sixteen octets [10].

   Implementations SHOULD NOT populate ID payload with IP addresses due
   to interoperability issues such as problems with Network Address
   Translator (NAT) traversal, and problems with IP verification
   behavior.

   Deployments may only want to consider using the IP address as ID if
   all of the following are true:

   o the peer's IP address is static, not dynamically changing

   o the peer is NOT behind a NAT'ing device

o the administrator intends the implementation to verify that the
   peer source address matches the IP address in the ID received, and
   that in the iPAddress field in the peer certificate's
   SubjectAltName extension.

Implementations MUST be capable of verifying that the IP address
presented in ID matches via bitwise comparison the IP address present
in the certificate's iPAddress field of the SubjectAltName extension.
Implementations MUST perform this verification by default. When
comparing the contents of ID with the iPAddress field in the
SubjectAltName extension for equality, binary comparison MUST be
performed. Note that certificates may contain multiple address
identity types -- in which case, at least one must match the source
IP. If the default is enabled, then a mismatch between the two
addresses MUST be treated as an error, and security association setup
MUST be aborted. This event SHOULD be auditable. Implementations
MAY provide a configuration option to (i.e., local policy
configuration can enable) skip that verification step, but that
option MUST be off by default. We include the "option-to-skip-
validation" in order to permit better interoperability as current
implementations vary greatly in how they behave on this topic.

In addition, implementations MUST be capable of verifying that the
address contained in the ID is the same as the address contained in
the IP header. Implementations SHOULD be able to check the address
in either the outermost or innermost IP header and MAY provide a
configuration option for specifying which is to be checked. If there
is no configuration option provided, an implementation SHOULD check
the peer source address contained in the outermost header (as is the
practice of most of today's implementations). If ID is one of the IP
address types, then implementations MUST perform this verification by
default. If this default is enabled, then a mismatch MUST be treated
as an error, and security association setup MUST be aborted. This
event SHOULD be auditable. Implementations MAY provide a
configuration option to (i.e. local policy configuration can enable)
skip that verification step, but that option MUST be off by default.
We include the "option-to-skip-validation" in order to permit better
interoperability, as current implementations vary greatly in how they
behave on the topic of verification of source IP.

If the default for both the verifications above are enabled, then, by
transitive property, the implementation will also be verifying that
the peer source IP address matches via a bitwise comparison the
contents of the iPAddress field in the SubjectAltName extension in
the certificate. In addition, implementations MAY allow
administrators to configure a local policy that explicitly requires
that the peer source IP address match via a bitwise comparison the
contents of the iPAddress field in the SubjectAltName extension in

the certificate. Implementations SHOULD allow administrators to
configure a local policy that skips this validation check.

Implementations MAY support substring, wildcard, or regular
expression matching of the contents of ID to look up the policy in
the SPD, and such would be a matter of local security policy
configuration.

Implementations MAY use the IP address found in the header of packets
received from the peer to look up the policy, but such
implementations MUST still perform verification of the ID payload.
Although packet IP addresses are inherently untrustworthy and must
therefore be independently verified, it is often useful to use the
apparent IP address of the peer to locate a general class of policies
that will be used until the mandatory identity-based policy lookup
can be performed.

For instance, if the IP address of the peer is unrecognized, a VPN
gateway device might load a general "road warrior" policy that
specifies a particular Certification Authority (CA) that is trusted
to issue certificates that contain a valid rfc822Name, which can be
used by that implementation to perform authorization based on access
control lists (ACLs) after the peer's certificate has been validated.
The rfc822Name can then be used to determine the policy that provides
specific authorization to access resources (such as IP addresses,
ports, and so forth).

As another example, if the IP address of the peer is recognized to be
a known peer VPN endpoint, policy may be determined using that
address, but until the identity (address) is validated by validating
the peer certificate, the policy MUST NOT be used to authorize any
IPsec traffic.

3.1.2. ID_FQDN

Implementations MUST support the ID_FQDN ID type, generally to
support host-based access control lists for hosts without fixed IP
addresses. However, implementations SHOULD NOT use the DNS to map
the FQDN to IP addresses for input into any policy decisions, unless
that mapping is known to be secure, for example, if DNSSEC [11] were
employed for that FQDN.

If ID contains an ID_FQDN, implementations MUST be capable of
verifying that the identity contained in the ID payload matches
identity information contained in the peer end-entity certificate, in
the dNSName field in the SubjectAltName extension. Implementations
MUST perform this verification by default. When comparing the
contents of ID with the dNSName field in the SubjectAltName extension

for equality, case-insensitive string comparison MUST be performed.
Note that case-insensitive string comparison works on
internationalized domain names (IDNs) as well (See IDN [12]).
Substring, wildcard, or regular expression matching MUST NOT be
performed for this comparison. If this default is enabled, then a
mismatch MUST be treated as an error, and security association setup
MUST be aborted. This event SHOULD be auditable. Implementations
MAY provide a configuration option to (i.e., local policy
configuration can enable) skip that verification step, but that
option MUST be off by default. We include the "option-to-skip-
validation" in order to permit better interoperability, as current
implementations vary greatly in how they behave on this topic.

Implementations MAY support substring, wildcard, or regular
expression matching of the contents of ID to look up the policy in
the SPD, and such would be a matter of local security policy
configuration.

3.1.3. ID_USER_FQDN

Implementations MUST support the ID_USER_FQDN ID type, generally to
support user-based access control lists for users without fixed IP
addresses. However, implementations SHOULD NOT use the DNS to map
the FQDN portion to IP addresses for input into any policy decisions,
unless that mapping is known to be secure, for example, if DNSSEC
[11] were employed for that FQDN.

Implementations MUST be capable of verifying that the identity
contained in the ID payload matches identity information contained in
the peer end-entity certificate, in the rfc822Name field in the
SubjectAltName extension. Implementations MUST perform this
verification by default. When comparing the contents of ID with the
rfc822Name field in the SubjectAltName extension for equality, case-
insensitive string comparison MUST be performed. Note that case-
insensitive string comparison works on internationalized domain names
(IDNs) as well (See IDN [12]). Substring, wildcard, or regular
expression matching MUST NOT be performed for this comparison. If
this default is enabled, then a mismatch MUST be treated as an error,
and security association setup MUST be aborted. This event SHOULD be
auditable. Implementations MAY provide a configuration option to
(i.e., local policy configuration can enable) skip that verification
step, but that option MUST be off by default. We include the
"option-to-skip-validation" in order to permit better
interoperability, as current implementations vary greatly in how they
behave on this topic.

Implementations MAY support substring, wildcard, or regular
expression matching of the contents of ID to look up policy in the
SPD, and such would be a matter of local security policy
configuration.

3.1.4. ID_IPV4_ADDR_SUBNET, ID_IPV6_ADDR_SUBNET, ID_IPV4_ADDR_RANGE, ID_IPV6_ADDR_RANGE

Note that RFC 3779 [13] defines blocks of addresses using the
certificate extension identified by:

        id-pe-ipAddrBlock OBJECT IDENTIFIER ::= { id-pe 7 }

although use of this extension in IKE is considered experimental at
this time.

3.1.5. ID_DER_ASN1_DN

Implementations MUST support receiving the ID_DER_ASN1_DN ID type.
Implementations MUST be capable of generating this type, and the
decision to do so will be a matter of local security policy
configuration. When generating this type, implementations MUST
populate the contents of ID with the Subject field from the end-
entity certificate, and MUST do so such that a binary comparison of
the two will succeed. If there is not a match, this MUST be treated
as an error, and security association setup MUST be aborted. This
event SHOULD be auditable.

Implementations MUST NOT populate ID with the Subject from the end-
entity certificate if it is empty, even though an empty certificate
Subject is explicitly allowed in the "Subject" section of the PKIX
certificate profile.

Regarding SPD matching, implementations MUST be able to perform
matching based on a bitwise comparison of the entire DN in ID to its
entry in the SPD. However, operational experience has shown that
using the entire DN in local configuration is difficult, especially
in large-scale deployments. Therefore, implementations also MUST be
able to perform SPD matches of any combination of one or more of the
C, CN, O, OU attributes within Subject DN in the ID to the same in
the SPD. Implementations MAY support matching using additional DN
attributes in any combination, although interoperability is far from
certain and is dubious. Implementations MAY also support performing
substring, wildcard, or regular expression matches for any of its
supported DN attributes from ID, in any combination, to the SPD.
Such flexibility allows deployers to create one SPD entry on the
gateway for an entire department of a company (e.g., O=Foobar Inc.,
OU=Engineering) while still allowing them to draw out other details

from the DN (e.g., CN=John Doe) for auditing purposes. All the above
is a matter of local implementation and local policy definition and
enforcement capability, not bits on the wire, but will have a great
impact on interoperability.

3.1.6. ID_DER_ASN1_GN

Implementations MUST NOT generate this type, because the recipient
will be unlikely to know how to use it.

3.1.7. ID_KEY_ID

The ID_KEY_ID type used to specify pre-shared keys and thus is out of
scope.

3.1.8. Selecting an Identity from a Certificate

Implementations MUST support certificates that contain more than a
single identity, such as when the Subject field and the
SubjectAltName extension are both populated, or the SubjectAltName
extension contains multiple identities irrespective of whether or not
the Subject is empty. In many cases, a certificate will contain an
identity, such as an IP address, in the SubjectAltName extension in
addition to a non-empty Subject.

Implementations should populate ID with whichever identity is likely
to be named in the peer's policy. In practice, this generally means
FQDN, or USER_FQDN, but this information may also be available to the
administrator through some out-of-band means. In the absence of such
out-of-band configuration information, the identity with which an
implementation chooses to populate the ID payload is a local matter.

3.1.9. Subject for DN Only

If an FQDN is intended to be processed as an identity for the
purposes of ID matching, it MUST be placed in the dNSName field of
the SubjectAltName extension. Implementations MUST NOT populate the
Subject with an FQDN in place of populating the dNSName field of the
SubjectAltName extension.

While nothing prevents an FQDN, USER_FQDN, or IP address information
from appearing somewhere in the Subject contents, such entries MUST
NOT be interpreted as identity information for the purposes of
matching with ID or for policy lookup.

3.1.10. Binding Identity to Policy

In the presence of certificates that contain multiple identities,
implementations should select the most appropriate identity from the
certificate and populate the ID with that. The recipient MUST use
the identity sent as a first key when selecting the policy. The
recipient MUST also use the most specific policy from that database
if there are overlapping policies caused by wildcards (or the
implementation can de-correlate the policy database so there will not
be overlapping entries, or it can also forbid creation of overlapping
policies and leave the de-correlation process to the administrator,
but, as this moves the problem to the administrator, it is NOT
RECOMMENDED).

For example, imagine that an implementation is configured with a
certificate that contains both a non-empty Subject and a dNSName.
The sender's policy may specify which of those to use, and it
indicates the policy to the other end by sending that ID. If the
recipient has both a specific policy for the dNSName for this host
and generic wildcard rule for some attributes present in the Subject
field, it will match a different policy depending on which ID is
sent. As the sender knows why it wanted to connect the peer, it also
knows what identity it should use to match the policy it needs to the
operation it tries to perform; it is the only party who can select
the ID adequately.

In the event that the policy cannot be found in the recipient's SPD
using the ID sent, then the recipient MAY use the other identities in
the certificate when attempting to match a suitable policy. For
example, say the certificate contains a non-empty Subject field, a
dNSName and an iPAddress. If an iPAddress is sent in ID but no
specific entry exists for the address in the policy database, the
recipient MAY search in the policy database based on the Subject or
the dNSName contained in the certificate.

3.2. Certificate Request Payload

The Certificate Request (CERTREQ) Payload allows an implementation to
request that a peer provide some set of certificates or certificate
revocation lists (CRLs). It is not clear from ISAKMP exactly how
that set should be specified or how the peer should respond. We
describe the semantics on both sides.

3.2.1. Certificate Type

   The Certificate Type field identifies to the peer the type of
   certificate keying materials that are desired. ISAKMP defines 10
   types of Certificate Data that can be requested and specifies the
   syntax for these types. For the purposes of this document, only the
   following types are relevant:

      o X.509 Certificate - Signature
      o Revocation Lists (CRL and ARL)
      o PKCS #7 wrapped X.509 certificate

   The use of the other types are out of the scope of this document:

      o X.509 Certificate - Key Exchange
      o PGP (Pretty Good Privacy) Certificate
      o DNS Signed Key
      o Kerberos Tokens
      o SPKI (Simple Public Key Infrastructure) Certificate
      o X.509 Certificate Attribute

3.2.2. X.509 Certificate - Signature

   This type requests that the end-entity certificate be a certificate
   used for signing.

3.2.3. Revocation Lists (CRL and ARL)

   ISAKMP does not support Certificate Payload sizes over approximately
   64K, which is too small for many CRLs, and UDP fragmentation is
   likely to occur at sizes much smaller than that. Therefore, the
   acquisition of revocation material is to be dealt with out-of-band of
   IKE. For this and other reasons, implementations SHOULD NOT generate
   CERTREQs where the Certificate Type is "Certificate Revocation List
   (CRL)" or "Authority Revocation List (ARL)". Implementations that do
   generate such CERTREQs MUST NOT require the recipient to respond with
   a CRL or ARL, and MUST NOT fail when not receiving any. Upon receipt
   of such a CERTREQ, implementations MAY ignore the request.

   In lieu of exchanging revocation lists in-band, a pointer to
   revocation checking SHOULD be listed in either the
   CRLDistributionPoints (CDP) or the AuthorityInfoAccess (AIA)
   certificate extensions (see Section 5 for details). Unless other
   methods for obtaining revocation information are available,
   implementations SHOULD be able to process these attributes, and from
   them be able to identify cached revocation material, or retrieve the
   relevant revocation material from a URL, for validation processing.
   In addition, implementations MUST have the ability to configure

validation checking information for each certification authority.
Regardless of the method (CDP, AIA, or static configuration), the
acquisition of revocation material SHOULD occur out-of-band of IKE.
Note, however, that an inability to access revocation status data
through out-of-band means provides a potential security vulnerability
that could potentially be exploited by an attacker.

3.2.4. PKCS #7 wrapped X.509 certificate

This ID type defines a particular encoding (not a particular
certificate type); some current implementations may ignore CERTREQs
they receive that contain this ID type, and the editors are unaware
of any implementations that generate such CERTREQ messages.
Therefore, the use of this type is deprecated. Implementations
SHOULD NOT require CERTREQs that contain this Certificate Type.
Implementations that receive CERTREQs that contain this ID type MAY
treat such payloads as synonymous with "X.509 Certificate -
Signature".

3.2.5. Location of Certificate Request Payloads

In IKEv1 Main Mode, the CERTREQ payload MUST be in messages 4 and 5.

3.2.6. Presence or Absence of Certificate Request Payloads

When in-band exchange of certificate keying materials is desired,
implementations MUST inform the peer of this by sending at least one
CERTREQ. In other words, an implementation that does not send any
CERTREQs during an exchange SHOULD NOT expect to receive any CERT
payloads.

3.2.7. Certificate Requests

3.2.7.1. Specifying Certification Authorities

When requesting in-band exchange of keying materials, implementations
SHOULD generate CERTREQs for every peer trust anchor that local
policy explicitly deems trusted during a given exchange.
Implementations SHOULD populate the Certification Authority field
with the Subject field of the trust anchor, populated such that
binary comparison of the Subject and the Certification Authority will
succeed.

Upon receipt of a CERTREQ, implementations MUST respond by sending at least the end-entity certificate corresponding to the Certification Authority listed in the CERTREQ unless local security policy configuration specifies that keying materials must be exchanged out-of-band. Implementations MAY send certificates other than the end-entity certificate (see Section 3.3 for discussion).

Note that, in the case where multiple end-entity certificates may be available that chain to different trust anchors, implementations SHOULD resort to local heuristics to determine which trust anchor is most appropriate to use for generating the CERTREQ. Such heuristics are out of the scope of this document.

3.2.7.2. Empty Certification Authority Field

Implementations SHOULD generate CERTREQs where the Certificate Type is "X.509 Certificate - Signature" and where the Certification Authority field is not empty. However, implementations MAY generate CERTREQs with an empty Certification Authority field under special conditions. Although PKIX prohibits certificates with an empty Issuer field, there does exist a use case where doing so is appropriate, and carries special meaning in the IKE context. This has become a convention within the IKE interoperability tests and usage space, and so its use is specified, explained here for the sake of interoperability.

USE CASE: Consider the rare case where you have a gateway with multiple policies for a large number of IKE peers: some of these peers are business partners, some are remote-access employees, some are teleworkers, some are branch offices, and/or the gateway may be simultaneously serving many customers (e.g., Virtual Routers). The total number of certificates, and corresponding trust anchors, is very high -- say, hundreds. Each of these policies is configured with one or more acceptable trust anchors, so that in total, the gateway has one hundred (100) trust anchors that could possibly used to authenticate an incoming connection. Assume that many of those connections originate from hosts/gateways with dynamically assigned IP addresses, so that the source IP of the IKE initiator is not known to the gateway, nor is the identity of the initiator (until it is revealed in Main Mode message 5). In IKE main mode message 4, the responder gateway will need to send a CERTREQ to the initiator. Given this example, the gateway will have no idea which of the hundred possible Certification Authorities to send in the CERTREQ. Sending all possible Certification Authorities will cause significant processing delays, bandwidth consumption, and UDP fragmentation, so this tactic is ruled out.

In such a deployment, the responder gateway implementation should be able to do all it can to indicate a Certification Authority in the CERTREQ. This means the responder SHOULD first check SPD to see if it can match the source IP, and find some indication of which CA is associated with that IP. If this fails (because the source IP is not familiar, as in the case above), then the responder SHOULD have a configuration option specifying which CAs are the default CAs to indicate in CERTREQ during such ambiguous connections (e.g., send CERTREQ with these N CAs if there is an unknown source IP). If such a fall-back is not configured or impractical in a certain deployment scenario, then the responder implementation SHOULD have both of the following configuration options:

o   send a CERTREQ payload with an empty Certification Authority
    field, or

o   terminate the negotiation with an appropriate error message and
    audit log entry.

Receiving a CERTREQ payload with an empty Certification Authority field indicates that the recipient should send all/any end-entity certificates it has, regardless of the trust anchor. The initiator should be aware of what policy and which identity it will use, as it initiated the connection on a matched policy to begin with, and can thus respond with the appropriate certificate.

If, after sending an empty CERTREQ in Main Mode message 4, a responder receives a certificate in message 5 that chains to a trust anchor that the responder either (a) does NOT support, or (b) was not configured for the policy (that policy was now able to be matched due to having the initiator's certificate present), this MUST be treated as an error, and security association setup MUST be aborted. This event SHOULD be auditable.

Instead of sending an empty CERTREQ, the responder implementation MAY be configured to terminate the negotiation on the grounds of a conflict with locally configured security policy.

The decision of which to configure is a matter of local security policy; this document RECOMMENDS that both options be presented to administrators.

More examples and explanation of this issue are included in "More on Empty CERTREQs" (Appendix B).

3.2.8. Robustness

3.2.8.1. Unrecognized or Unsupported Certificate Types

   Implementations MUST be able to deal with receiving CERTREQs with
   unsupported Certificate Types. Absent any recognized and supported
   CERTREQ types, implementations MAY treat them as if they are of a
   supported type with the Certification Authority field left empty,
   depending on local policy. ISAKMP [2] Section 5.10, "Certificate
   Request Payload Processing", specifies additional processing.

3.2.8.2. Undecodable Certification Authority Fields

   Implementations MUST be able to deal with receiving CERTREQs with
   undecodable Certification Authority fields. Implementations MAY
   ignore such payloads, depending on local policy. ISAKMP specifies
   other actions which may be taken.

3.2.8.3. Ordering of Certificate Request Payloads

   Implementations MUST NOT assume that CERTREQs are ordered in any way.

3.2.9. Optimizations

3.2.9.1. Duplicate Certificate Request Payloads

   Implementations SHOULD NOT send duplicate CERTREQs during an
   exchange.

3.2.9.2. Name Lowest 'Common' Certification Authorities

   When a peer's certificate keying material has been cached, an
   implementation can send a hint to the peer to elide some of the
   certificates the peer would normally include in the response. In
   addition to the normal set of CERTREQs that are sent specifying the
   trust anchors, an implementation MAY send CERTREQs specifying the
   relevant cached end-entity certificates. When sending these hints,
   it is still necessary to send the normal set of trust anchor CERTREQs
   because the hints do not sufficiently convey all of the information
   required by the peer. Specifically, either the peer may not support
   this optimization or there may be additional chains that could be
   used in this context but will not be if only the end-entity
   certificate is specified.

   No special processing is required on the part of the recipient of
   such a CERTREQ, and the end-entity certificates will still be sent.
   On the other hand, the recipient MAY elect to elide certificates
   based on receipt of such hints.

CERTREQs must contain information that identifies a Certification Authority certificate, which results in the peer always sending at least the end-entity certificate. Always sending the end-entity certificate allows implementations to determine unambiguously when a new certificate is being used by a peer (perhaps because the previous certificate has just expired), which may result in a failure because a new intermediate CA certificate might not be available to validate the new end-entity certificate). Implementations that implement this optimization MUST recognize when the end-entity certificate has changed and respond to it by not performing this optimization if the exchange must be retried so that any missing keying materials will be sent during retry.

3.2.9.3. Example

Imagine that an IKEv1 implementation has previously received and cached the peer certificate chain TA->CA1->CA2->EE. If, during a subsequent exchange, this implementation sends a CERTREQ containing the Subject field in certificate TA, this implementation is requesting that the peer send at least three certificates: CA1, CA2, and EE. On the other hand, if this implementation also sends a CERTREQ containing the Subject field of CA2, the implementation is providing a hint that only one certificate needs to be sent: EE. Note that in this example, the fact that TA is a trust anchor should not be construed to imply that TA is a self-signed certificate.

3.3. Certificate Payload

The Certificate (CERT) Payload allows the peer to transmit a single certificate or CRL. Multiple certificates should be transmitted in multiple payloads. For backwards-compatibility reasons, implementations MAY send intermediate CA certificates in addition to the appropriate end-entity certificate(s), but SHOULD NOT send any CRLs, ARLs, or trust anchors. Exchanging trust anchors and especially CRLs and ARLs in IKE would increase the likelihood of UDP fragmentation, make the IKE exchange more complex, and consume additional network bandwidth.

Note, however, that while the sender of the CERT payloads SHOULD NOT send any certificates it considers trust anchors, it's possible that the recipient may consider any given intermediate CA certificate to be a trust anchor. For instance, imagine the sender has the certificate chain TA1->CA1->EE1 while the recipient has the certificate chain TA2->EE2 where TA2=CA1. The sender is merely including an intermediate CA certificate, while the recipient receives a trust anchor.

However, not all certificate forms that are legal in the PKIX
certificate profile make sense in the context of IPsec. The issue of
how to represent IKE-meaningful name-forms in a certificate is
especially problematic. This document provides a profile for a
subset of the PKIX certificate profile that makes sense for IKEv1/
ISAKMP.

3.3.1. Certificate Type

The Certificate Type field identifies to the peer the type of
certificate keying materials that are included. ISAKMP defines 10
types of Certificate Data that can be sent and specifies the syntax
for these types. For the purposes of this document, only the
following types are relevant:

   o X.509 Certificate - Signature
   o Revocation Lists (CRL and ARL)
   o PKCS #7 wrapped X.509 certificate

The use of the other types are out of the scope of this document:

   o X.509 Certificate - Key Exchange
   o PGP Certificate
   o DNS Signed Key
   o Kerberos Tokens
   o SPKI Certificate
   o X.509 Certificate Attribute

3.3.2. X.509 Certificate - Signature

This type specifies that Certificate Data contains a certificate used
for signing.

3.3.3. Revocation Lists (CRL and ARL)

These types specify that Certificate Data contains an X.509 CRL or
ARL. These types SHOULD NOT be sent in IKE. See Section 3.2.3 for
discussion.

3.3.4. PKCS #7 Wrapped X.509 Certificate

This type defines a particular encoding, not a particular certificate
type. Implementations SHOULD NOT generate CERTs that contain this
Certificate Type. Implementations SHOULD accept CERTs that contain
this Certificate Type because several implementations are known to
generate them. Note that those implementations sometimes include

entire certificate hierarchies inside a single CERT PKCS #7 payload,
which violates the requirement specified in ISAKMP that this payload
contain a single certificate.

3.3.5. Location of Certificate Payloads

In IKEv1 Main Mode, the CERT payload MUST be in messages 5 and 6.

3.3.6. Certificate Payloads Not Mandatory

An implementation that does not receive any CERTREQs during an
exchange SHOULD NOT send any CERT payloads, except when explicitly
configured to proactively send CERT payloads in order to interoperate
with non-compliant implementations that fail to send CERTREQs even
when certificates are desired. In this case, an implementation MAY
send the certificate chain (not including the trust anchor)
associated with the end-entity certificate. This MUST NOT be the
default behavior of implementations.

Implementations whose local security policy configuration expects
that a peer must receive certificates through out-of-band means
SHOULD ignore any CERTREQ messages that are received. Such a
condition has been known to occur due to non-compliant or buggy
implementations.

Implementations that receive CERTREQs from a peer that contain only
unrecognized Certification Authorities MAY elect to terminate the
exchange, in order to avoid unnecessary and potentially expensive
cryptographic processing, denial-of-service (resource starvation)
attacks.

3.3.7. Response to Multiple Certification Authority Proposals

In response to multiple CERTREQs that contain different Certification
Authority identities, implementations MAY respond using an end-entity
certificate which chains to a CA that matches any of the identities
provided by the peer.

3.3.8. Using Local Keying Materials

Implementations MAY elect to skip parsing or otherwise decoding a
given set of CERTs if those same keying materials are available via
some preferable means, such as the case where certificates from a
previous exchange have been cached.

3.3.9. Multiple End-Entity Certificates

Implementations SHOULD NOT send multiple end-entity certificates and recipients SHOULD NOT be expected to iterate over multiple end-entity certificates.

If multiple end-entity certificates are sent, they MUST have the same public key; otherwise, the responder does not know which key was used in the Main Mode message 5.

3.3.10. Robustness

3.3.10.1. Unrecognized or Unsupported Certificate Types

Implementations MUST be able to deal with receiving CERTs with unrecognized or unsupported Certificate Types. Implementations MAY discard such payloads, depending on local policy. ISAKMP [2] Section 5.10, "Certificate Request Payload Processing", specifies additional processing.

3.3.10.2. Undecodable Certificate Data Fields

Implementations MUST be able to deal with receiving CERTs with undecodable Certificate Data fields. Implementations MAY discard such payloads, depending on local policy. ISAKMP specifies other actions that may be taken.

3.3.10.3. Ordering of Certificate Payloads

Implementations MUST NOT assume that CERTs are ordered in any way.

3.3.10.4. Duplicate Certificate Payloads

Implementations MUST support receiving multiple identical CERTs during an exchange.

3.3.10.5. Irrelevant Certificates

Implementations MUST be prepared to receive certificates and CRLs that are not relevant to the current exchange. Implementations MAY discard such extraneous certificates and CRLs.

Implementations MAY send certificates that are irrelevant to an exchange. One reason for including certificates that are irrelevant to an exchange is to minimize the threat of leaking identifying information in exchanges where CERT is not encrypted in IKEv1. It should be noted, however, that this probably provides rather poor protection against leaking the identity.

Another reason for including certificates that seem irrelevant to an
exchange is that there may be two chains from the Certification
Authority to the end entity, each of which is only valid with certain
validation parameters (such as acceptable policies). Since the end-
entity doesn't know which parameters the relying party is using, it
should send the certificates needed for both chains (even if there's
only one CERTREQ).

Implementations SHOULD NOT send multiple end-entity certificates and
recipients SHOULD NOT be expected to iterate over multiple end-entity
certificates.

3.3.11. Optimizations

3.3.11.1. Duplicate Certificate Payloads

Implementations SHOULD NOT send duplicate CERTs during an exchange.
Such payloads should be suppressed.

3.3.11.2. Send Lowest 'Common' Certificates

When multiple CERTREQs are received that specify certification
authorities within the end-entity certificate chain, implementations
MAY send the shortest chain possible. However, implementations
SHOULD always send the end-entity certificate. See Section 3.2.9.2
for more discussion of this optimization.

3.3.11.3. Ignore Duplicate Certificate Payloads

Implementations MAY employ local means to recognize CERTs that have
already been received and SHOULD discard these duplicate CERTs.

3.3.11.4. Hash Payload

IKEv1 specifies the optional use of the Hash Payload to carry a
pointer to a certificate in either of the Phase 1 public key
encryption modes. This pointer is used by an implementation to
locate the end-entity certificate that contains the public key that a
peer should use for encrypting payloads during the exchange.

Implementations SHOULD include this payload whenever the public
portion of the keypair has been placed in a certificate.

4. Use of Certificates in RFC 4301 and IKEv2

4.1. Identification Payload

The Peer Authorization Database (PAD) as described in RFC 4301 [14]
describes the use of the ID payload in IKEv2 and provides a formal
model for the binding of identity to policy in addition to providing
services that deal more specifically with the details of policy
enforcement, which are generally out of scope of this document. The
PAD is intended to provide a link between the SPD and the security
association management in protocols such as IKE. See RFC 4301 [14],
Section 4.4.3 for more details.

Note that IKEv2 adds an optional IDr payload in the second exchange
that the initiator may send to the responder in order to specify
which of the responder's multiple identities should be used. The
responder MAY choose to send an IDr in the third exchange that
differs in type or content from the initiator-generated IDr. The
initiator MUST be able to receive a responder-generated IDr that is a
different type from the one the initiator generated.

4.2. Certificate Request Payload

4.2.1. Revocation Lists (CRL and ARL)

IKEv2 does not support Certificate Payload sizes over approximately
64K. See Section 3.2.3 for the problems this can cause.

4.2.1.1. IKEv2's Hash and URL of X.509 certificate

This ID type defines a request for the peer to send a hash and URL of
its X.509 certificate, instead of the actual certificate itself.
This is a particularly useful mechanism when the peer is a device
with little memory and lower bandwidth, e.g., a mobile handset or
consumer electronics device.

If the IKEv2 implementation supports URL lookups, and prefers such a
URL to receiving actual certificates, then the implementation will
want to send a notify of type HTTP_CERT_LOOKUP_SUPPORTED. From IKEv2
[3], Section 3.10.1, "This notification MAY be included in any
message that can include a CERTREQ payload and indicates that the
sender is capable of looking up certificates based on an HTTP-based
URL (and hence presumably would prefer to receive certificate
specifications in that format)". If an HTTP_CERT_LOOKUP_SUPPORTED
notification is sent, the sender MUST support the http scheme. See
Section 4.3.1 for more discussion of HTTP_CERT_LOOKUP_SUPPORTED.

4.2.1.2. Location of Certificate Request Payloads

   In IKEv2, the CERTREQ payload must be in messages 2 and 3. Note that
   in IKEv2, it is possible to have one side authenticating with
   certificates while the other side authenticates with pre-shared keys.

4.3. Certificate Payload

4.3.1. IKEv2's Hash and URL of X.509 Certificate

   This type specifies that Certificate Data contains a hash and the URL
   to a repository where an X.509 certificate can be retrieved.

   An implementation that sends an HTTP_CERT_LOOKUP_SUPPORTED
   notification MUST support the http scheme and MAY support the ftp
   scheme, and MUST NOT require any specific form of the url-path, and
   it SHOULD support having user-name, password, and port parts in the
   URL. The following are examples of mandatory forms:

   o http://certs.example.com/certificate.cer
   o http://certs.example.com/certs/cert.pl?u=foo;a=pw;valid-to=+86400
   o http://certs.example.com/%0a/../foo/bar/zappa

   while the following is an example of a form that SHOULD be supported:

   o http://user:password@certs.example.com:8888/certificate.cer

   FTP MAY be supported, and if it is, the following is an example of
   the ftp scheme that MUST be supported:

   o ftp://ftp.example.com/pub/certificate.cer

4.3.2. Location of Certificate Payloads

   In IKEv2, the CERT payload MUST be in messages 3 and 4. Note that in
   IKEv2, it is possible to have one side authenticating with
   certificates while the other side authenticates with pre-shared keys.

4.3.3. Ordering of Certificate Payloads

   For IKEv2, implementations MUST NOT assume that any but the first
   CERT is ordered in any way. IKEv2 specifies that the first CERT
   contain an end-entity certificate that can be used to authenticate
   the peer.

5. Certificate Profile for IKEv1/ISAKMP and IKEv2

Except where specifically stated in this document, implementations
MUST conform to the requirements of the PKIX [5] certificate profile.

5.1. X.509 Certificates

Users deploying IKE and IPsec with certificates have often had little
control over the capabilities of CAs available to them.
Implementations of this specification may include configuration knobs
to disable checks required by this specification in order to permit
use with inflexible and/or noncompliant CAs. However, all checks on
certificates exist for a specific reason involving the security of
the entire system. Therefore, all checks MUST be enabled by default.
Administrators and users ought to understand the security purpose for
the various checks, and be clear on what security will be lost by
disabling the check.

5.1.1. Versions

Although PKIX states that "implementations SHOULD be prepared to
accept any version certificate", in practice, this profile requires
certain extensions that necessitate the use of Version 3 certificates
for all but self-signed certificates used as trust anchors.
Implementations that conform to this document MAY therefore reject
Version 1 and Version 2 certificates in all other cases.

5.1.2. Subject

Certification Authority implementations MUST be able to create
certificates with Subject fields with at least the following four
attributes: CN, C, O, and OU. Implementations MAY support other
Subject attributes as well. The contents of these attributes SHOULD
be configurable on a certificate-by-certificate basis, as these
fields will likely be used by IKE implementations to match SPD
policy.

See Section 3.1.5 for details on how IKE implementations need to be
able to process Subject field attributes for SPD policy lookup.

5.1.2.1. Empty Subject Name

IKE Implementations MUST accept certificates that contain an empty
Subject field, as specified in the PKIX certificate profile.
Identity information in such certificates will be contained entirely
in the SubjectAltName extension.

5.1.2.2. Specifying Hosts and not FQDN in the Subject Name

   Implementations that desire to place host names that are not intended
   to be processed by recipients as FQDNs (for instance "Gateway
   Router") in the Subject MUST use the commonName attribute.

5.1.2.3. EmailAddress

   As specified in the PKIX certificate profile, implementations MUST
   NOT populate X.500 distinguished names with the emailAddress
   attribute.

5.1.3. X.509 Certificate Extensions

   Conforming IKE implementations MUST recognize extensions that must or
   may be marked critical according to this specification. These
   extensions are: KeyUsage, SubjectAltName, and BasicConstraints.

   Certification Authority implementations SHOULD generate certificates
   such that the extension criticality bits are set in accordance with
   the PKIX certificate profile and this document. With respect to
   compliance with the PKIX certificate profile, IKE implementations
   processing certificates MAY ignore the value of the criticality bit
   for extensions that are supported by that implementation, but MUST
   support the criticality bit for extensions that are not supported by
   that implementation. That is, a relying party SHOULD processes all
   the extensions it is aware of whether the bit is true or false -- the
   bit says what happens when a relying party cannot process an
   extension.

```
         implements    bit in cert    PKIX mandate    behavior
         ----------------------------------------------------
         yes           true           true            ok
         yes           true           false           ok or reject
         yes           false          true            ok or reject
         yes           false          false           ok
         no            true           true            reject
         no            true           false           reject
         no            false          true            reject
         no            false          false           ok
```

5.1.3.1. AuthorityKeyIdentifier and SubjectKeyIdentifier

   Implementations SHOULD NOT assume support for the
   AuthorityKeyIdentifier or SubjectKeyIdentifier extensions. Thus,
   Certification Authority implementations should not generate
   certificate hierarchies that are overly complex to process in the
   absence of these extensions, such as those that require possibly

verifying a signature against a large number of similarly named CA
certificates in order to find the CA certificate that contains the
key that was used to generate the signature.

5.1.3.2. KeyUsage

IKE uses an end-entity certificate in the authentication process.
The end-entity certificate may be used for multiple applications. As
such, the CA can impose some constraints on the manner that a public
key ought to be used. The KeyUsage (KU) and ExtendedKeyUsage (EKU)
extensions apply in this situation.

Since we are talking about using the public key to validate a
signature, if the KeyUsage extension is present, then at least one of
the digitalSignature or the nonRepudiation bits in the KeyUsage
extension MUST be set (both can be set as well). It is also fine if
other KeyUsage bits are set.

A summary of the logic flow for peer cert validation follows:

o If no KU extension, continue.

o If KU present and doesn't mention digitalSignature or
   nonRepudiation (both, in addition to other KUs, is also fine),
   reject cert.

o If none of the above, continue.

5.1.3.3. PrivateKeyUsagePeriod

The PKIX certificate profile recommends against the use of this
extension. The PrivateKeyUsageExtension is intended to be used when
signatures will need to be verified long past the time when
signatures using the private keypair may be generated. Since IKE
security associations (SAs) are short-lived relative to the intended
use of this extension in addition to the fact that each signature is
validated only a single time, the usefulness of this extension in the
context of IKE is unclear. Therefore, Certification Authority
implementations MUST NOT generate certificates that contain the
PrivateKeyUsagePeriod extension. If an IKE implementation receives a
certificate with this set, it SHOULD ignore it.

5.1.3.4. CertificatePolicies

   Many IKE implementations do not currently provide support for the
   CertificatePolicies extension. Therefore, Certification Authority
   implementations that generate certificates that contain this
   extension SHOULD NOT mark the extension as critical. As is the case
   with all certificate extensions, a relying party receiving this
   extension but who can process the extension SHOULD NOT reject the
   certificate because it contains the extension.

5.1.3.5. PolicyMappings

   Many IKE implementations do not support the PolicyMappings extension.
   Therefore, implementations that generate certificates that contain
   this extension SHOULD NOT mark the extension as critical.

5.1.3.6. SubjectAltName

   Deployments that intend to use an ID of FQDN, USER_FQDN, IPV4_ADDR,
   or IPV6_ADDR MUST issue certificates with the corresponding
   SubjectAltName fields populated with the same data. Implementations
   SHOULD generate only the following GeneralName choices in the
   SubjectAltName extension, as these choices map to legal IKEv1/ISAKMP/
   IKEv2 Identification Payload types: rfc822Name, dNSName, or
   iPAddress. Although it is possible to specify any GeneralName choice
   in the Identification Payload by using the ID_DER_ASN1_GN ID type,
   implementations SHOULD NOT assume support for such functionality, and
   SHOULD NOT generate certificates that do so.

5.1.3.6.1. dNSName

   If the IKE ID type is FQDN, then this field MUST contain a fully
   qualified domain name. If the IKE ID type is FQDN, then the dNSName
   field MUST match its contents. Implementations MUST NOT generate
   names that contain wildcards. Implementations MAY treat certificates
   that contain wildcards in this field as syntactically invalid.

   Although this field is in the form of an FQDN, IKE implementations
   SHOULD NOT assume that this field contains an FQDN that will resolve
   via the DNS, unless this is known by way of some out-of-band
   mechanism. Such a mechanism is out of the scope of this document.
   Implementations SHOULD NOT treat the failure to resolve as an error.

5.1.3.6.2. iPAddress

If the IKE ID type is IPV4_ADDR or IPV6_ADDR, then the iPAddress
field MUST match its contents. Note that although PKIX permits CIDR
[15] notation in the "Name Constraints" extension, the PKIX
certificate profile explicitly prohibits using CIDR notation for
conveying identity information. In other words, the CIDR notation
MUST NOT be used in the SubjectAltName extension.

5.1.3.6.3. rfc822Name

If the IKE ID type is USER_FQDN, then the rfc822Name field MUST match
its contents. Although this field is in the form of an Internet mail
address, IKE implementations SHOULD NOT assume that this field
contains a valid email address, unless this is known by way of some
out-of-band mechanism. Such a mechanism is out of the scope of this
document.

5.1.3.7. IssuerAltName

Certification Authority implementations SHOULD NOT assume that other
implementations support the IssuerAltName extension, and especially
should not assume that information contained in this extension will
be displayed to end users.

5.1.3.8. SubjectDirectoryAttributes

The SubjectDirectoryAttributes extension is intended to convey
identification attributes of the subject. IKE implementations MAY
ignore this extension when it is marked non-critical, as the PKIX
certificate profile mandates.

5.1.3.9. BasicConstraints

The PKIX certificate profile mandates that CA certificates contain
this extension and that it be marked critical. IKE implementations
SHOULD reject CA certificates that do not contain this extension.
For backwards compatibility, implementations may accept such
certificates if explicitly configured to do so, but the default for
this setting MUST be to reject such certificates.

5.1.3.10. NameConstraints

Many IKE implementations do not support the NameConstraints
extension. Since the PKIX certificate profile mandates that this
extension be marked critical when present, Certification Authority
implementations that are interested in maximal interoperability for
IKE SHOULD NOT generate certificates that contain this extension.

5.1.3.11. PolicyConstraints

   Many IKE implementations do not support the PolicyConstraints
   extension. Since the PKIX certificate profile mandates that this
   extension be marked critical when present, Certification Authority
   implementations that are interested in maximal interoperability for
   IKE SHOULD NOT generate certificates that contain this extension.

5.1.3.12. ExtendedKeyUsage

   The CA SHOULD NOT include the ExtendedKeyUsage (EKU) extension in
   certificates for use with IKE. Note that there were three IPsec-
   related object identifiers in EKU that were assigned in 1999. The
   semantics of these values were never clearly defined. The use of
   these three EKU values in IKE/IPsec is obsolete and explicitly
   deprecated by this specification. CAs SHOULD NOT issue certificates
   for use in IKE with them. (For historical reference only, those
   values were id-kp-ipsecEndSystem, id-kp-ipsecTunnel, and id-kp-
   ipsecUser.)

   The CA SHOULD NOT mark the EKU extension in certificates for use with
   IKE and one or more other applications. Nevertheless, this document
   defines an ExtendedKeyUsage keyPurposeID that MAY be used to limit a
   certificate's use:

   id-kp-ipsecIKE OBJECT IDENTIFIER ::= { id-kp 17 }

   where id-kp is defined in RFC 3280 [5]. If a certificate is intended
   to be used with both IKE and other applications, and one of the other
   applications requires use of an EKU value, then such certificates
   MUST contain either the keyPurposeID id-kp-ipsecIKE or
   anyExtendedKeyUsage [5], as well as the keyPurposeID values
   associated with the other applications. Similarly, if a CA issues
   multiple otherwise-similar certificates for multiple applications
   including IKE, and it is intended that the IKE certificate NOT be
   used with another application, the IKE certificate MAY contain an EKU
   extension listing a keyPurposeID of id-kp-ipsecIKE to discourage its
   use with the other application. Recall, however, that EKU extensions
   in certificates meant for use in IKE are NOT RECOMMENDED.

   Conforming IKE implementations are not required to support EKU. If a
   critical EKU extension appears in a certificate and EKU is not
   supported by the implementation, then RFC 3280 requires that the
   certificate be rejected. Implementations that do support EKU MUST
   support the following logic for certificate validation:

   o If no EKU extension, continue.

   o If EKU present AND contains either id-kp-ipsecIKE or
      anyExtendedKeyUsage, continue.

   o Otherwise, reject cert.

5.1.3.13. CRLDistributionPoints

   Because this document deprecates the sending of CRLs in-band, the use
   of CRLDistributionPoints (CDP) becomes very important if CRLs are
   used for revocation checking (as opposed to, say, Online Certificate
   Status Protocol - OCSP [16]). The IPsec peer either needs to have a
   URL for a CRL written into its local configuration, or it needs to
   learn it from CDP. Therefore, Certification Authority
   implementations SHOULD issue certificates with a populated CDP.

   Failure to validate the CRLDistributionPoints/
   IssuingDistributionPoint pair can result in CRL substitution where an
   entity knowingly substitutes a known good CRL from a different
   distribution point for the CRL that is supposed to be used, which
   would show the entity as revoked. IKE implementations MUST support
   validating that the contents of CRLDistributionPoints match those of
   the IssuingDistributionPoint to prevent CRL substitution when the
   issuing CA is using them. At least one CA is known to default to
   this type of CRL use. See Section 5.2.2.5 for more information.

   CDPs SHOULD be "resolvable". Several non-compliant Certification
   Authority implementations are well known for including unresolvable
   CDPs like http://localhost/path_to_CRL and http:///path_to_CRL that
   are equivalent to failing to include the CDP extension in the
   certificate.

   See the IETF IPR Web page for CRLDistributionPoints intellectual
   property rights (IPR) information. Note that both the
   CRLDistributionPoints and IssuingDistributionPoint extensions are
   RECOMMENDED but not REQUIRED by the PKIX certificate profile, so
   there is no requirement to license any IPR.

5.1.3.14. InhibitAnyPolicy

   Many IKE implementations do not support the InhibitAnyPolicy
   extension. Since the PKIX certificate profile mandates that this
   extension be marked critical when present, Certification Authority
   implementations that are interested in maximal interoperability for
   IKE SHOULD NOT generate certificates that contain this extension.

5.1.3.15. FreshestCRL

   IKE implementations MUST NOT assume that the FreshestCRL extension
   will exist in peer certificates. Note that most IKE implementations
   do not support delta CRLs.

5.1.3.16. AuthorityInfoAccess

   The PKIX certificate profile defines the AuthorityInfoAccess
   extension, which is used to indicate "how to access CA information
   and services for the issuer of the certificate in which the extension
   appears". Because this document deprecates the sending of CRLs in-
   band, the use of AuthorityInfoAccess (AIA) becomes very important if
   OCSP [16] is to be used for revocation checking (as opposed to CRLs).
   The IPsec peer either needs to have a URI for the OCSP query written
   into its local configuration, or it needs to learn it from AIA.
   Therefore, implementations SHOULD support this extension, especially
   if OCSP will be used.

5.1.3.17. SubjectInfoAccess

   The PKIX certificate profile defines the SubjectInfoAccess
   certificate extension, which is used to indicate "how to access
   information and services for the subject of the certificate in which
   the extension appears". This extension has no known use in the
   context of IPsec. Conformant IKE implementations SHOULD ignore this
   extension when present.

5.2. X.509 Certificate Revocation Lists

   When validating certificates, IKE implementations MUST make use of
   certificate revocation information, and SHOULD support such
   revocation information in the form of CRLs, unless non-CRL revocation
   information is known to be the only method for transmitting this
   information. Deployments that intend to use CRLs for revocation
   SHOULD populate the CRLDistributionPoints extension. Therefore,
   Certification Authority implementations MUST support issuing
   certificates with this field populated. IKE implementations MAY
   provide a configuration option to disable use of certain types of
   revocation information, but that option MUST be off by default. Such
   an option is often valuable in lab testing environments.

5.2.1. Multiple Sources of Certificate Revocation Information

   IKE implementations that support multiple sources of obtaining
   certificate revocation information MUST act conservatively when the
   information provided by these sources is inconsistent: when a
   certificate is reported as revoked by one trusted source, the
   certificate MUST be considered revoked.

5.2.2. X.509 Certificate Revocation List Extensions

5.2.2.1. AuthorityKeyIdentifier

   Certification Authority implementations SHOULD NOT assume that IKE
   implementations support the AuthorityKeyIdentifier extension, and
   thus should not generate certificate hierarchies which are overly
   complex to process in the absence of this extension, such as those
   that require possibly verifying a signature against a large number of
   similarly named CA certificates in order to find the CA certificate
   which contains the key that was used to generate the signature.

5.2.2.2. IssuerAltName

   Certification Authority implementations SHOULD NOT assume that IKE
   implementations support the IssuerAltName extension, and especially
   should not assume that information contained in this extension will
   be displayed to end users.

5.2.2.3. CRLNumber

   As stated in the PKIX certificate profile, all issuers MUST include
   this extension in all CRLs.

5.2.2.4. DeltaCRLIndicator

5.2.2.4.1. If Delta CRLs Are Unsupported

   IKE implementations that do not support delta CRLs MUST reject CRLs
   that contain the DeltaCRLIndicator (which MUST be marked critical
   according to the PKIX certificate profile) and MUST make use of a
   base CRL if it is available. Such implementations MUST ensure that a
   delta CRL does not "overwrite" a base CRL, for instance, in the
   keying material database.

5.2.2.4.2. Delta CRL Recommendations

   Since some IKE implementations that do not support delta CRLs may
   behave incorrectly or insecurely when presented with delta CRLs,
   administrators and deployers should consider whether issuing delta
   CRLs increases security before issuing such CRLs. And, if all the
   elements in the VPN and PKI systems do not adequately support Delta
   CRLs, then their use should be questioned.

   The editors are aware of several implementations that behave in an
   incorrect or insecure manner when presented with delta CRLs. See
   Appendix A for a description of the issue. Therefore, this
   specification RECOMMENDS NOT issuing delta CRLs at this time. On the
   other hand, failure to issue delta CRLs may expose a larger window of
   vulnerability if a full CRL is not issued as often as delta CRLs
   would be. See the Security Considerations section of the PKIX [5]
   certificate profile for additional discussion. Implementers as well
   as administrators are encouraged to consider these issues.

5.2.2.5. IssuingDistributionPoint

   A CA that is using CRLDistributionPoints may do so to provide many
   "small" CRLs, each only valid for a particular set of certificates
   issued by that CA. To associate a CRL with a certificate, the CA
   places the CRLDistributionPoints extension in the certificate, and
   places the IssuingDistributionPoint in the CRL. The
   distributionPointName field in the CRLDistributionPoints extension
   MUST be identical to the distributionPoint field in the
   IssuingDistributionPoint extension. At least one CA is known to
   default to this type of CRL use. See Section 5.1.3.13 for more
   information.

5.2.2.6. FreshestCRL

   Given the recommendations against Certification Authority
   implementations generating delta CRLs, this specification RECOMMENDS
   that implementations do not populate CRLs with the FreshestCRL
   extension, which is used to obtain delta CRLs.

5.3. Strength of Signature Hashing Algorithms

   At the time that this document is being written, popular
   certification authorities and CA software issue certificates using
   the RSA-with-SHA1 and RSA-with-MD5 signature algorithms.
   Implementations MUST be able to validate certificates with either of
   those algorithms.

As described in [17], both the MD5 and SHA-1 hash algorithms are
weaker than originally expected with respect to hash collisions.
Certificates that use these hash algorithms as part of their
signature algorithms could conceivably be subject to an attack where
a CA issues a certificate with a particular identity, and the
recipient of that certificate can create a different valid
certificate with a different identity. So far, such an attack is
only theoretical, even with the weaknesses found in the hash
algorithms.

Because of the recent attacks, there has been a heightened interest
in having widespread deployment of additional signature algorithms.
The algorithm that has been mentioned most often is RSA-with-SHA256,
two types of which are described in detail in [18]. It is widely
expected that this signature algorithm will be much more resilient to
collision-based attacks than the current RSA-with-SHA1 and RSA-with-
MD5, although no proof of that has been shown. There is active
discussion in the cryptographic community of better hash functions
that could be used in signature algorithms.

In order to interoperate, all implementations need to be able to
validate signatures for all algorithms that the implementations will
encounter. Therefore, implementations SHOULD be able to use
signatures that use the sha256WithRSAEncryption signature algorithm
(PKCS#1 version 1.5) as soon as possible. At the time that this
document is being written, there is at least one CA that supports
generating certificates with sha256WithRSAEncryption signature
algorithm, and it is expected that there will be significant
deployment of this algorithm by the end of 2007.

6. Configuration Data Exchange Conventions

Below, we present a common format for exchanging configuration data.
Implementations MUST support these formats, MUST support receiving
arbitrary whitespace at the beginning and end of any line, MUST
support receiving arbitrary line lengths although they SHOULD
generate lines less than 76 characters, and MUST support receiving
the following three line-termination disciplines: LF (US-ASCII 10),
CR (US-ASCII 13), and CRLF.

6.1. Certificates

Certificates MUST be Base64 [19] encoded and appear between the
following delimiters:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

6.2. CRLs and ARLs

CRLs and ARLs MUST be Base64 encoded and appear between the following
delimiters:

```
-----BEGIN CRL-----
-----END CRL-----
```

6.3. Public Keys

IKE implementations MUST support two forms of public keys:
certificates and so-called "raw" keys. Certificates should be
transferred in the same form as Section 6.1. A raw key is only the
SubjectPublicKeyInfo portion of the certificate, and MUST be Base64
encoded and appear between the following delimiters:

```
-----BEGIN PUBLIC KEY-----
-----END PUBLIC KEY-----
```

6.4. PKCS#10 Certificate Signing Requests

A PKCS#10 [9] Certificate Signing Request MUST be Base64 encoded and
appear between the following delimiters:

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

7. Security Considerations

7.1. Certificate Request Payload

The Contents of CERTREQ are not encrypted in IKE. In some
environments, this may leak private information. Administrators in
some environments may wish to use the empty Certification Authority
option to prevent such information from leaking, at the cost of
performance.

7.2. IKEv1 Main Mode

Certificates may be included in any message, and therefore
implementations may wish to respond with CERTs in a message that
offers privacy protection in Main Mode messages 5 and 6.

Implementations may not wish to respond with CERTs in the second
message, thereby violating the identity protection feature of Main
Mode in IKEv1.

7.3. Disabling Certificate Checks

   It is important to note that anywhere this document suggests
   implementers provide users with the configuration option to simplify,
   modify, or disable a feature or verification step, there may be
   security consequences for doing so. Deployment experience has shown
   that such flexibility may be required in some environments, but
   making use of such flexibility can be inappropriate in others. Such
   configuration options MUST default to "enabled" and it is appropriate
   to provide warnings to users when disabling such features.

8. Acknowledgements

   The authors would like to acknowledge the expired document "A PKIX
   Profile for IKE" (July 2000) for providing valuable materials for
   this document.

   The authors would like to especially thank Eric Rescorla, one of its
   original authors, in addition to Greg Carter, Steve Hanna, Russ
   Housley, Charlie Kaufman, Tero Kivinen, Pekka Savola, Paul Hoffman,
   and Gregory Lebovitz for their valuable comments, some of which have
   been incorporated verbatim into this document. Paul Knight performed
   the arduous task of converting the text to XML format.

9. References

9.1. Normative References

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

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

   [3]   Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC
          4306, December 2005.

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

   [5]   Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509
          Public Key Infrastructure Certificate and Certificate
          Revocation List (CRL) Profile", RFC 3280, April 2002.

   [6]   Piper, D., "The Internet IP Security Domain of Interpretation
          for ISAKMP", RFC 2407, November 1998.

   [7]   Bradner, S., "Key words for use in RFCs to Indicate Requirement
         Levels", BCP 14, RFC 2119, March 1997.

   [8]   Postel, J., "Internet Protocol", STD 5, RFC 791, September
         1981.

   [9]   Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request
         Syntax Specification Version 1.7", RFC 2986, November 2000.

9.2. Informative References

   [10] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6)
         Specification", RFC 2460, December 1998.

   [11] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose,
         "DNS Security Introduction and Requirements", RFC 4033, March
         2005.

   [12] Faltstrom, P., Hoffman, P., and A. Costello,
         "Internationalizing Domain Names in Applications (IDNA)", RFC
         3490, March 2003.

   [13] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP
         Addresses and AS Identifiers", RFC 3779, June 2004.

   [14] Kent, S. and K. Seo, "Security Architecture for the Internet
         Protocol", RFC 4301, December 2005.

   [15] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR):
         The Internet Address Assignment and Aggregation Plan", BCP 122,
         RFC 4632, August 2006.

   [16] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams,
         "X.509 Internet Public Key Infrastructure Online Certificate
         Status Protocol - OCSP", RFC 2560, June 1999.

   [17] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes
         in Internet Protocols", RFC 4270, November 2005.

   [18] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms
         and Identifiers for RSA Cryptography for use in the Internet
         X.509 Public Key Infrastructure Certificate and Certificate
         Revocation List (CRL) Profile", RFC 4055, June 2005.

   [19] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings",
         RFC 4648, October 2006.

Appendix A.   The Possible Dangers of Delta CRLs

   The problem is that the CRL processing algorithm is sometimes written
   incorrectly with the assumption that all CRLs are base CRLs and it is
   assumed that CRLs will pass content validity tests. Specifically,
   such implementations fail to check the certificate against all
   possible CRLs: if the first CRL that is obtained from the keying
   material database fails to decode, no further revocation checks are
   performed for the relevant certificate. This problem is compounded
   by the fact that implementations that do not understand delta CRLs
   may fail to decode such CRLs due to the critical DeltaCRLIndicator
   extension. The algorithm that is implemented in this case is
   approximately:

   o fetch newest CRL

   o check validity of CRL signature

   o if CRL signature is valid, then

   o if CRL does not contain unrecognized critical extensions and
      certificate is on CRL, then set certificate status to revoked

   The authors note that a number of PKI toolkits do not even provide a
   method for obtaining anything but the newest CRL, which in the
   presence of delta CRLs may in fact be a delta CRL, not a base CRL.

   Note that the above algorithm is dangerous in many ways. See the
   PKIX [5] certificate profile for the correct algorithm.

Appendix B.   More on Empty CERTREQs

   Sending empty certificate requests is commonly used in
   implementations, and in the IPsec interop meetings, vendors have
   generally agreed that it means that send all/any end-entity
   certificates you have (if multiple end-entity certificates are sent,
   they must have same public key, as otherwise, the other end does not
   know which key was used). For 99% of cases, the client has exactly
   one certificate and public key, so it really doesn't matter, but the
   server might have multiple; thus, it simply needs to say to the
   client, use any certificate you have. If we are talking about
   corporate VPNs, etc., even if the client has multiple certificates or
   keys, all of them would be usable when authenticating to the server,
   so the client can simply pick one.

   If there is some real difference on which certificate to use (like
   ones giving different permissions), then the client must be
   configured anyway, or it might even ask the user which one to use

(the user is the only one who knows whether he needs admin
privileges, thus needs to use admin cert, or if the normal email
privileges are ok, thus uses email only cert).

In 99% of the cases, the client has exactly one certificate, so it
will send it. In 90% of the rest of the cases, any of the
certificates is ok, as they are simply different certificates from
the same CA, or from different CAs for the same corporate VPN, thus
any of them is ok.

Sending empty certificate requests has been agreed there to mean
"give me your cert, any cert".

Justification:

o Responder first does all it can to send a CERTREQ with a CA, check
   for IP match in SPD, have a default set of CAs to use in ambiguous
   cases, etc.

o Sending empty CERTREQs is fairly common in implementations today,
   and is generally accepted to mean "send me a certificate, any
   certificate that works for you".

o Saves responder sending potentially hundreds of certs, the
   fragmentation problems that follow, etc.

o In +90% of use cases, Initiators have exactly one certificate.

o In +90% of the remaining use cases, the multiple certificates it
   has are issued by the same CA.

o In the remaining use case(s) -- if not all the others above -- the
   Initiator will be configured explicitly with which certificate to
   send, so responding to an empty CERTREQ is easy.

The following example shows why initiators need to have sufficient
policy definition to know which certificate to use for a given
connection it initiates.

EXAMPLE: Your client (initiator) is configured with VPN policies for
gateways A and B (representing perhaps corporate partners).

The policies for the two gateways look something like:

```
Acme Company policy (gateway A)
   Engineering can access 10.1.1.0
           Trusted CA: CA-A, Trusted Users: OU=Engineering
     Partners can access 20.1.1.0
           Trusted CA: CA-B, Trusted Users: OU=AcmePartners

Bizco Company policy (gateway B)
   Sales can access 30.1.1.0
           Trusted CA: CA-C, Trusted Users: OU=Sales
     Partners can access 40.1.1.0
           Trusted CA: CA-B, Trusted Users: OU=BizcoPartners
```

You are an employee of Acme and you are issued the following certificates:

o From CA-A: CN=JoeUser,OU=Engineering
o From CA-B: CN=JoePartner,OU=BizcoPartners

The client MUST be configured locally to know which CA to use when connecting to either gateway. If your client is not configured to know the local credential to use for the remote gateway, this scenario will not work either. If you attempt to connect to Bizco, everything will work... as you are presented with responding with a certificate signed by CA-B or CA-C... as you only have a certificate from CA-B you are OK. If you attempt to connect to Acme, you have an issue because you are presented with an ambiguous policy selection. As the initiator, you will be presented with certificate requests from both CA-A and CA-B. You have certificates issued by both CAs, but only one of the certificates will be usable. How does the client know which certificate it should present? It must have sufficiently clear local policy specifying which one credential to present for the connection it initiates.

Author's Address

   Brian Korver
   Network Resonance, Inc.
   2483 E. Bayshore Rd.
   Palo Alto, CA 94303
   US

   Phone: +1 650 812 7705
   EMail: briank@networkresonance.com

Full Copyright Statement

   Copyright (C) The IETF Trust (2007).

   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.

   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, THE IETF TRUST 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.

Intellectual Property

   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.

Acknowledgement

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