Syntaxes for Unambiguous Identification of Certificates and Public Keys
draft-ietf-pkix-certid-keyid-01

Abstract

This document discusses methods to unambiguously identify a particular certificate and a particular public key that may or may not be in a certificate. This document further proposes standardizing on one of these structures on a going-forward basis throughout new PKIX standards.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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 9, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of
1. Introduction

When working with X.509 certificates and other cryptographic structures, it is frequently desirable to refer to another certificate, or to a raw public key, directly and unambiguously. The "gold standard" for certificates is to refer to the certificate’s issuer name and serial number. Furthermore, [RFC5280] and [X.509] define a subject key identifier extension, which provides a means of identifying distinct keys used by the same subject.

Unfortunately, neither the issuer name and serial number, nor the subject key identifier, are guaranteed to be universally unique. Multiple CA certificates can share the same subject name. The subject key identifier is meant to facilitate path construction [RFC5280], but is only unique to the extent that the certificate issuer guarantees it to be unique within some domain. While still being compliant with RFC 5280, it is possible to create subject key identifiers in different certificates that refer to the same subject public key. It is also possible for agents to ignore or reuse subject key identifiers with impunity, and to use those identifiers for interchange.

This document provides a review of the existing ASN.1 structures to unambiguously identify a particular certificate and a particular public key. This document further proposes standardizing on a single certificate identifier structure on a going-forward basis throughout new PKIX standards.

1.1. Requirements Language

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.

1.2. Errata

This document may be fit for Standards Track or Best Current Practice (BCP) status, depending on input from the Internet community. This draft is under consideration for PKIX WG adoption; it is not yet a PKIX WG document.
2. Existing Structures for Identifying Certificates

At least four structures are used throughout IETF documents to identify a certificate uniquely: ESSCertID [RFC2634], CertID from OCSP [RFC2560], ESSCertIDv2 [RFC5035], and SCVPCertID [RFC5055]. These structures use the same basic techniques: digest the target with a cryptographic hash algorithm, and optionally refer to the target using standardized naming. Using this method, an implementation can cryptographically verify that the right certificate or public key is chosen, but can also use standard methods to locate the certificate or public key if the search facility does not index by the particular hash.

Each of these structures, however, uses slightly different methods from each other to do the same thing, resulting in a proliferation of implementations when only one is needed. Furthermore, some of these structures lack algorithm agility, and therefore are susceptible to attack when practical collisions can be found in the underlying cryptographic algorithms.

3. Uniform Structure for Identifying Certificates

In order to reduce the variance of implementations, this document proposes standardizing on a single structure for uniquely identifying certificates on a going-forward basis: ESSCertIDv2 from RFC 5035, with additional clarifying semantics discussed below. In this document, ESSCertIDv2 is aliased to PKIXCertID. Future new standards SHOULD define and use PKIXCertID when attempting to identify a specific, existing certificate. Future revisions of existing standards can continue to use their old structures for backwards-compatibility purposes.

PKIXCertID ::= ESSCertIDv2
ESSCertIDv2 is reproduced below for convenience.

```
ESSCertIDv2 ::= SEQUENCE {
  hashAlgorithm AlgorithmIdentifier
    DEFAULT {algorithm id-sha256},
  certHash Hash,
  issuerSerial IssuerSerial OPTIONAL
}
```

Hash ::= OCTET STRING

IssuerSerial ::= SEQUENCE {
  issuer GeneralNames,
  serialNumber CertificateSerialNumber
}

The fields of PKIXCertID (ESSCertIDv2, as used here) are defined as follows.

hashAlgorithm contains the identifier of the algorithm used in computing certHash.

certHash is computed over the entire DER-encoded certificate, including the signature, tag, and length octets, using the algorithm specified in hashAlgorithm.

issuerSerial holds the identification of the certificate. issuerSerial is optional. If present, it MAY be used to facilitate lookup of the certificate. Additionally, an implementation MAY use issuerSerial to eliminate a certificate from consideration early if the implementation does not support the certificate type.

The issuerSerial field is normative if it is present. Specifically, if a certificate is found that matches the issuerSerial, but does not match the hash, an implementation MUST skip or reject that certificate, because it does not match the hash. If a certificate is found that matches the hash, but does not match the issuerSerial, an implementation MUST skip or reject that certificate, because it does not match the issuerSerial. The assumption is that the implementation that generated identifier will have the certificate and will actually process it in order to create a proper issuerSerial for it.

The semantics of issuerSerial are the same as those in RFC 5035.

The rationale for choosing ESSCertIDv2 is based on several considerations. The ordering seems optimal: an implementation that cannot process a particular hash algorithm is best served by figuring
out the hash algorithm first, then skipping the rest of the structure if it cannot work with that hash algorithm. The default algorithm, SHA-256, is a reasonable choice on a going-forward basis and maintains compatibility with RFC 5035. (Note that this specification makes no requirement to support any particular hash algorithm; specifying the DEFAULT merely saves octets on the wire.) Finally, IssuerSerial is not strictly limited to X.509 public key certificates; other certificates such as attribute certificates can be identified.

This structure has a similar structure in attribute certificates, except that the attribute certificate profile’s IssuerSerial structure includes an optional issuerUID UniqueIdentifier OPTIONAL at the end [RFC5755]. The structure in this specification explicitly does not include issuerUID for several reasons. First, compatibility with RFC 5035 is desired. Second, UniqueIdentifier fields are properties of version 2 certificates, which are rarely seen in practice. Third, UniqueIdentifier fields were seen as insufficient to solve the identification problems with certificates, since any CA can put anything in a UniqueIdentifier field (thus making their uniqueness guarantees suspect).

New protocol designers need to decide first whether it is necessary to uniquely identify an existing certificate, or whether it is sufficient to identify certificates non-uniquely. If uniqueness is required, then PKIXCertID SHOULD [TBD: MUST] be used. If uniqueness is not required, another matching method is appropriate, such as issuer and serial number, subject name, and so forth.

4. Uniform Structures for Identifying Public Keys

Unlike the CertID structure, there is not a proliferation of structures (among the various standards surveyed) to unambiguously identify a public key. The argument against providing such a method is mainly that PKIX works with certified keys, not with "raw" keys.

In fact there are at least two common structures for identifying keys. One can use the SubjectPublicKeyInfo field to actually specify a public key: this technique is the equivalent of identifying a public key by its value. To refer to a public key by reference, one can use the ObjectDigestInfo structure of attribute certificates, as specified in RFC 5755, to identify a public key by its hash. When used in context, these structures can refer to "raw" keys, or can refer to all certificates that contain the keys (hence ceritified keys). The remainder of this section discusses how to use ObjectDigestInfo to refer to a public key by reference.
ObjectDigestInfo ::= SEQUENCE {
  digestedObjectType  ENUMERATED {
    publicKey            (0),
    publicKeyCert        (1),
    otherObjectTypes     (2) },
  otherObjectTypeID   OBJECT IDENTIFIER OPTIONAL,
  digestAlgorithm     AlgorithmIdentifier,
  objectDigest        BIT STRING
}

The ObjectDigestInfo field is reproduced above for convenience. When identifying a public key by reference, the ObjectDigestInfo field’s digestedObjectType is set to publicKey(0), otherObjectTypeID is ABSENT, and the digestAlgorithm and objectDigest identify the hash algorithm and hash of the public key’s DER-encoded SubjectPublicKeyInfo representation. Note that the SubjectPublicKeyInfo in a certificate if the public key algorithm uses parameters inherited through a CA’s public key certificate. For details, see Section 7.3 of [RFC5755].

Using ObjectDigestInfo alone could be insufficient for a recipient to look up a public key. In such a circumstance, the ObjectDigestInfo MAY be placed in a Holder field of [RFC5755]. The baseCertificateID of Holder MAY be used to look up a certificate containing the partial or complete SubjectPublicKeyInfo. In this case, the baseCertificateID is advisory in nature, because the intent of the sender is to identify a key or all certificates containing a key, not a particular certificate. A failure MUST NOT be generated merely because an implementation cannot find a certificate matching baseCertificateID.

Generating a public key identifier that is a public key by value (SubjectPublicKeyInfo) is useful when the public key is sufficiently small, and a strong hash is either similarly-sized or larger. Referring to the public key by value will also save time compared to looking up the public key from a database.

It is worth noting that a public key identifier can only refer to a public key of an asymmetric keypair; it MUST NOT refer to a symmetric key or to an asymmetric private key. The Security Considerations section provides further discussion.
5. Acknowledgements

The author thanks Russ Housley and Jim Schaad for direction and guidance during the development of this document.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

The PKIXCertID and public key identifier structures discussed here provide integrity checking of the named certificate or key, but are not themselves integrity-checked. An implementer SHOULD use these structures within other integrity-checked structures, such as within X.509 public key or attribute certificates.

Weak hash algorithms will limit the utility of these structures. An implementation MUST skip or reject certificates or keys when the cryptographic integrity of the hash function is compromised. On the other hand, an advanced algorithm might not be broadly compatible with all possible relying parties. Implementers SHOULD be aware of this tension when choosing a hash algorithm.

A public key identifier MUST NOT refer to a symmetric key or to an asymmetric private key. Referring to such keys by value (with SubjectPublicKeyInfo) would destroy the security properties of keeping such keys secret, since KeyID is intended for use in public documents such as certificates or other signed objects. Similar considerations apply to referring to such keys by reference (with ObjectDigestInfo). Additionally, RFC 5280 and RFC 3279 fix the semantics of SubjectPublicKeyInfo (and by extension, SubjectPublicKeyRef) to asymmetric public keys.

8. References

8.1. Normative References


8.2. Informative References


Appendix A. ASN.1 Modules

This Appendix will provide ASN.1 modules in a future revision.

A.1. 1988 ASN.1 Module

TBD
Author’s Address

Sean Leonard
Penango, Inc.
1215 K Street
17th Floor
Sacramento, CA  95814
USA

Email: dev+ietf@seantek.com
URI:  http://www.penango.com/