Additional Methods for Generating Subject Key Identifiers
and Subject Key Identifier Semantics Extension
draft-turner-additional-methods-4kis-07.txt

Abstract

This document specifies additional methods for generating Subject Key Identifiers (SKI). This document also specifies an extension to identify the algorithms used to generate the SKI.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

Copyright Notice

Copyright (c) 2012 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 publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

[RFC5280] defines the AKI (Authority Key Identifier) and SKI (Subject Key Identifier) certificate extensions. These extensions allow one certificate to refer to another certificate via the matching of these corresponding values. These identifiers enable a relying party to disambiguate between two CA (Certification Authority) certificates with the same Subject name, located in the same directory entry. These identifiers are used during certification path construction in support of heuristics to reduce relying party workload. These identifiers are not used during certificate path validation. These key identifiers are used by PKI-enabled security protocols, such as CMP (Certificate Management Protocol) [RFC4210] and CMS (Cryptographic Message Syntax) [RFC5652], to identify the certificate used to protect a message, a session, etc.

[RFC5280] describes two example mechanisms for generating AKI/SKI values: a 160-bit SHA-1 (Secure Hash Algorithm) hash of the public key and a four-bit type field with the value 0100 followed by the least significant 60 bits of the SHA-1 hash. Both of these mechanisms were designed to be non-security critical. That is, the use of a hash algorithm was intended to provide a high probability (but not a guarantee) of uniqueness. [RFC5280] allows for additional mechanisms. (This is consistent with the fact that the SKI and AKI extensions are always marked non-critical.) In addition, some security protocols (e.g., SMIME [RFC5751]) use key identifiers as a shorthand way to refer to a cert.

This document defines three additional mechanisms for generating SKI values, using SHA-256, SHA-384, and SHA-512 [SHS], that are similar to those examples defined in [RFC5280]. Sample code for SHA-256, SHA-384, and SHA-512 can be found in [RFC6234]. The motivation for defining these additional means of generating SKI values is to accommodate use of additional, standard one-way hash functions that are becoming more widely used in PKI contexts. Note that these example methods, like the examples methods from [RFC5280] are designed to be non-security critical.

With these additional mechanisms, CAs can omit code for algorithms that are otherwise unwanted or unused. For example, a CA that issues certificates hashed with SHA-256 and signed with ECDSA on the P-256 curve [RFC5480] might no longer need to implement SHA-1 as part of their CA application.

This document also defines an additional mechanism for generating an SKI value that hashes the public key algorithm identifier, in addition to the actual public key. This method is aligned with key identifiers defined in [ID.dane-protocol].
This document also specifies an extension to identify the algorithm used to generate the SKI.

1.1. Requirements Terminology

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 [RFC2119].

1.2. ASN.1

The extension is defined using ASN.1 [X.680], [X.681], [X.682], and [X.683].

2. Additional Methods for Generating Key Identifiers

As specified in [RFC5280], both authority and subject key identifiers SHOULD be derived from the public key. Four additional mechanisms CAs can use to identify public keys are as follows:

1) The keyIdentifier is composed of the leftmost 160-bits of the SHA-256 hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits).

2) The keyIdentifier is composed of the leftmost 160-bits of the SHA-384 hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits).

3) The keyIdentifier is composed of the leftmost 160-bits of the SHA-512 hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits).

4) The keyIdentifier is composed of the hash of the DER-encoding of the SubjectPublicKeyInfo value.

3. Subject Key Identifier Semantics Extension

The SKI semantics extension indicates the hash algorithm, the hash algorithm input used to compute the SKI, and any semantics the issuer chooses to communicate via the SKI. This allows the CA to embed additional semantics in to the SKI, allowing it to be used for purposes beyond certificate path building. This extension MAY, at the option of the certificate issuer, be either critical or non-critical. This extension is identified by id-pe-skiSemantics.

    ext-skiSemantics EXTENSION ::= {
        SYNTAX AlgorithmIdentifier { KI-ALGORITHM, {KIAlgs} }
        IDENTIFIED BY id-pe-skiSemantics }

Internet-Draft   Additional Methods For Key Identifier      July 3, 2012
This document defines the following set of algorithms:
- kialg-keyHash indicates that the key id is the, possibly truncated, hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits). The amount of truncation can be determined from the length of the actual key identifier. Truncation keeps the least significant bits of the hash. The required parameter of this algorithm identifies the hash algorithm that is used.

- kialg-4BitKeyHash indicates that the key id is a four-bit type field with the value 0100 followed the, possibly truncated, hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits). The amount of truncation can be determined from the length of the actual key identifier minus the first 4 bits. Truncation keeps the least significant bits of the hash. The required parameter of this algorithm identifies the hash algorithm that is used.

- kialg-keyInfoHash differs from kialg-keyHash in that the hash covers the key algorithm id in addition to the actual public key. kialg-keyInfoHash indicates that the key id is the, possibly truncated, hash of the subjectPublicKeyInfo field. The amount of truncation can be determined from the actual key identifier value. Truncation keeps the least significant bits of the hash. The required parameter of this algorithm identifies the hash algorithm that is used.

NOTE: If a CA wishes to indicate that it used the 2 example methods from [RFC5280], then kialg-keyHash and kialg-4BitKeyHash would use SHA-1 and in the latter case truncated the SHA-1 output to 60-bits.
4. Examples

This section provides some examples. The keys and SKIs are presented in hexadecimal (two hex digits per byte).

Given the following DER-encoded SubjectPublicKeyInfo value holding an P-256 ECDSA key:

\[
30 59 \\
30 13 \\
06 07 2A8648CE3D0201 -- id-ecPublicKey \\
06 08 2A8648CE3D030107 -- secp256r1 \\
03 42 00 \\
04 7F7F35A79794C950060B8029FC8F363A \\
28F11159692D934E6AC94819043534735 \\
F833B1A66652DC514337AFF7F5C9C75D \\
670C019D95A5D639B7274C64A9128BB
\]

The SHA-256 hash of the 65 bytes 047F7F...BB is:

\[
BF37B3E5808FD46D54B28E846311BCCE1CAD2E1A62AA9092EF3EB3F11451F44
\]

The SHA-1 hash of these 65 bytes is:

\[
6FEF9162C0A3F2E7608956D41C37DA0C8E87F0AE
\]

The SHA-256 hash of the 91 bytes 305930...BB is:

\[
6D20896AB88BD33B6B66554BD59B20225D8A75A296088148399D7BF763D57405
\]

EDITOR'S NOTE: The XXs in the following will be replaced with actual values once the OIDs have been assigned. We've assumed the extension OID will come from the PKIX id-pe arc and the method OIDs will come from the id-pkix arc.

Using method 1 from section 2, the subjectKeyIdentifier and ext-skiSemantics extensions would be:

\[
30 1D \\
06 03 551D0E -- id-ce-subjectKeyIdentifier \\
04 16 \\
04 14 BF37B3E5808FD46D54B28E846311BCCE1CAD2E1A
\]

\[
30 23 \\
06 08 2B0601050500701XX -- id-pe-skiSemantics \\
04 17 \\
30 15 \\
06 08 2B060105050717XX -- id-kialg-keyHash \\
06 09 608648016503040201 -- id-sha256
\]
Using the 1st method in [RFC5280], the two extensions would be:

30 1D
  06 03 551D0E -- id-ce-subjectKeyIdentifier
  04 16
    04 14 6FEF9162C0A3F2E7608956D41C37DA0C8E87F0AE

30 1F
  06 08 2B060105050701XX -- id-pe-skiSemantics
  04 13
    30 11
      06 08 2B060105050717XX -- id-kialg-keyHash
    06 05 2B0E03021A -- id-sha1

Using the 2nd method in [RFC5280], the two extensions would be:

30 11
  06 03 551D0E -- id-ce-subjectKeyIdentifier
  04 0A
    04 08 46FEF9162C0A3F2E

30 1F
  06 08 2B060105050701XX -- id-pe-skiSemantics
  04 13
    30 11
      06 08 2B060105050717XX -- id-kialg-keyHash
    06 05 2B0E03021A -- id-sha1

Using method 4 from section 2 with SHA-256 and no truncation, the two extensions would be:

30 26
  06 03 551D0E -- id-ce-subjectKeyIdentifier
  04 22
    04 20 6D20896AB8BD833B6B66554BD59B2022
    5D8A75A296088148399D7BF763D57405

30 23
  06 08 2B060105050701XX -- id-pe-skiSemantics
  04 17
    30 15
      06 08 2B060105050717XX -- id-kialg-keyInfoHash
    06 09 608648016503040201 -- id-sha256
5. Security Considerations


While hash algorithms provide preimage resistance, second-preimage resistance, and collision resistance, none of these properties are needed for key identifiers.

6. IANA Considerations

None.

NOTE there are some OIDs that need to be registered in the PKIX Arc. This will be completed later in the process.

7. Acknowledgements

The authors wish to thank Santosh Chokhani, Stephen Farrell, Tom Gindin, Peter Gutmann, Henry Holtz, David Kemp, Timothy Miller, Michael StJohns, Stefan Santesson, Jim Schaad, Rene Struik, Koichi Sugimoto, and Carl Wallace for taking the time to participate in the discussions about this document. The discussions resulted in numerous editorial and technical changes to the document.

8. References

8.1. Normative References


8.2. Informative References


Appendix A  ASN.1 Module

KISemantics-2012

{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-kiSemantics(TBD) }

DEFINITIONS EXPLICIT TAGS ::= BEGIN
IMPORTS

-- Imports are all from [RFC5912]

EXTENSION
FROM PKIX-CommonTypes-2009

{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkixCommon-02(57) }

id-pe, id-pkix
FROM PKIX1Explicit-2009

{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-explicit-02(51) }

AlgorithmIdentifier{}, DIGEST-ALGORITHM
FROM AlgorithmInformation-2009

{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-algorithmInformation-02(58) }

mda-sha1
FROM PKIXAlgs-2009

{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-algorithms2008-02(56) }

mda-sha224, mda-sha256, mda-sha384, mda-sha512
FROM PKIX1-PSS-OAEP-Algorithms-2009

{ iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-pkix1-rsa-pkalgs-02(54) } ;
ext-skiSemantics EXTENSION ::= {
  SYNTAX AlgorithmIdentifier { KI-ALGORITHM, {KIALgs} } 
  IDENTIFIED BY id-pe-skiSemantics }

id-pe-skiSemantics OBJECT IDENTIFIER ::= { id-pe TBD }

KIALgs KI-ALGORITHM ::= {
  kialg-keyHash |
  kialg-4BitKeyHash |
  kialg-keyInfoHash,
  ... }

KI-ALGORITHM ::= CLASS {
  &id OBJECT IDENTIFIER UNIQUE,
  &Params OPTIONAL
} WITH SYNTAX {
  IDENTIFIER &id
  [ PARAMS TYPE &Params ]
}

DigestAlgs DIGEST-ALGORITHM ::= {
  mda-sha1 |
  mda-sha256 |
  mda-sha384 |
  mda-sha512,
  ... }

kialg-keyHash KI-ALGORITHM ::= {
  IDENTIFIER id-kialg-keyHash
  PARAMS TYPE AlgorithmIdentifier {DIGEST-ALGORITHM, {DigestAlgs}}
}

id-kialg OBJECT IDENTIFIER ::= { id-pkix TBD }

id-kialg-keyHash OBJECT IDENTIFIER ::= { id-kialg TBD }

kialg-4BitKeyHash KI-ALGORITHM ::= {
  IDENTIFIER id-kialg-4BitKeyHash
  PARAMS TYPE AlgorithmIdentifier {DIGEST-ALGORITHM, {DigestAlgs}}
}

id-kialg-4BitKeyHash OBJECT IDENTIFIER ::= { id-kialg TBD }

kialg-keyInfoHash KI-ALGORITHM ::= {
  IDENTIFIER id-kialg-keyInfoHash
  PARAMS TYPE AlgorithmIdentifier {DIGEST-ALGORITHM, {DigestAlgs}}
}
id-kialg-keyInfoHash OBJECT IDENTIFIER ::= { id-kialg TBD }

Authors' Addresses

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
USA

EMail: turners@ieca.com

Stephen Kent
BBN Technologies
10 Moulton St.
Cambridge, MA 02138

EMail: kent@bbn.com

James Manger
Telstra
3 / 35 Collins Street
Melbourne, Victoria 3000
Australia

Email: james.h.manger@team.telstra.com