                  Hash Of Root Key Certificate Extension

Abstract

   This document specifies the Hash Of Root Key certificate extension.
   This certificate extension is carried in the self-signed certificate
   for a trust anchor, which is often called a Root Certification
   Authority (CA) certificate. This certificate extension unambiguously
   identifies the next public key that will be used at some point in the
   future as the next Root CA certificate, eventually replacing the
   current one.

Status of This Memo

   This document is not an Internet Standards Track specification; it is
   published for informational purposes.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Not all documents
   approved by the IESG are candidates for any level of Internet
   Standard; see Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8649.

Table of Contents

1. Introduction

   This document specifies the Hash Of Root Key X.509 version 3
   certificate extension. The extension is an optional addition to the
   Internet X.509 Public Key Infrastructure Certificate and Certificate
   Revocation List (CRL) Profile [RFC5280]. The certificate extension
   facilitates the orderly transition from one Root Certification
   Authority (CA) public key to the next. It does so by publishing the
   hash value of the next-generation public key in the current self-
   signed certificate. This hash value is a commitment to a particular
   public key in the next-generation self-signed certificate. This
   commitment allows a relying party to unambiguously recognize the
   next-generation self-signed certificate when it becomes available,
   install the new self-signed certificate in the trust anchor store,
   and eventually remove the previous one from the trust anchor store.

   A Root CA certificate MAY include the Hash Of Root Key certificate
   extension to provide the hash value of the next public key that will
   be used by the Root CA.

1.1. Terminology

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
   "OPTIONAL" in this document are to be interpreted as described in
   BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all
   capitals, as shown here.

1.2. ASN.1

   Certificates [RFC5280] use ASN.1 [X680]; Distinguished Encoding Rules
   (DER) [X690] are REQUIRED for certificate signing and validation.

2. Overview

   Before the initial deployment of the Root CA, the following are
   generated:

      R1 = The initial Root key pair
      R2 = The second-generation Root key pair
      H2 = Thumbprint (hash) of the public key of R2
      C1 = Self-signed certificate for R1, which also contains H2

   C1 is a self-signed certificate, and it contains H2 within the
   HashOfRootKey extension. C1 is distributed as part of the initial
   system deployment. The HashOfRootKey certificate extension is
   described in Section 3.

   When the time comes to replace the initial Root CA certificate, R1,
   the following are generated:

      R3 = The third-generation Root key pair
      H3 = Thumbprint (hash) the public key of R3
      C2 = Self-signed certificate for R2, which contains H3

   This is an iterative process. That is, R4 and H4 are generated when
   it is time for C3 to replace C2, and so on.

   The successor to the Root CA self-signed certificate can be delivered
   by any means. Whenever a new Root CA self-signed certificate is
   received, the recipient is able to verify that the potential Root CA
   certificate links back to a previously authenticated Root CA
   certificate with the HashOfRootKey certificate extension. That is,
   the recipient verifies the signature on the self-signed certificate
   and verifies that the hash of the DER-encoded SubjectPublicKeyInfo
   from the potential Root CA certificate matches the value from the
   HashOfRootKey certificate extension of the current Root CA
   certificate. Checking the self-signed certificate signature ensures
   that the certificate contains the subject name, public key algorithm
   identifier, and public key algorithm parameters intended by the key
   owner; these are important inputs to certification path validation as
   defined in Section 6 of [RFC5280]. Checking the hash of the
   SubjectPublicKeyInfo ensures that the certificate contains the
   intended public key. If either check fails, then the potential Root
   CA certificate is not a valid replacement, and the recipient
   continues to use the current Root CA certificate. If both checks

succeed, then the recipient adds the potential Root CA certificate to
the trust anchor store. As discussed in Section 5, the recipient can
remove the current Root CA certificate immediately in some
situations. In other situations, the recipient waits an appropriate
amount of time to ensure that existing certification paths continue
to validate.

3. Hash Of Root Key Certificate Extension

   The HashOfRootKey certificate extension MUST NOT be critical.

   The following ASN.1 [X680] [X690] syntax defines the HashOfRootKey
   certificate extension:

```
ext-HashOfRootKey EXTENSION ::= {    -- Only in Root CA certificates
   SYNTAX          HashedRootKey
   IDENTIFIED BY  id-ce-hashOfRootKey
   CRITICALITY    {FALSE} }

HashedRootKey ::= SEQUENCE {
   hashAlg         HashAlgorithm,       -- Hash algorithm used
   hashValue       OCTET STRING }       -- Hash of DER-encoded
                                        --   SubjectPublicKeyInfo

id-ce-hashOfRootKey  ::=  OBJECT IDENTIFIER { 1 3 6 1 4 1 51483 2 1 }
```

   The definitions of EXTENSION and HashAlgorithm can be found in
   [RFC5912].

   The hashAlg indicates the one-way hash algorithm that was used to
   compute the hash value.

   The hashValue contains the hash value computed from the next-
   generation public key. The public key is the DER-encoded
   SubjectPublicKeyInfo as defined in [RFC5280].

4. IANA Considerations

   This document has no IANA actions.

5. Operational Considerations

   Guidance on the transition from one root key to another is available
   in Section 4.4 of [RFC4210]. Of course, a root key is also known as
   a trust anchor. In particular, the oldWithNew and newWithOld advice
   ensures that relying parties are able to validate certificates issued
   under the current Root CA certificate and the next-generation Root CA
   certificate throughout the transition. The notAfter field in the

oldWithNew certificate MUST cover the validity period of all
unexpired certificates issued under the old Root CA private key.
Further, this advice SHOULD be followed by Root CAs to avoid the need
for all relying parties to make the transition at the same time.

After issuing the newWithOld certificate, the Root CA MUST stop using
the old private key to sign certificates.

Some enterprise and application-specific environments offer a
directory service or certificate repository to make certificate and
CRLs available to relying parties. Section 3 in [RFC5280] describes
a certificate repository. When a certificate repository is
available, the oldWithNew and newWithOld certificates SHOULD be
published before the successor to the current Root CA self-signed
certificate is released. Recipients that are able to obtain the
oldWithNew certificate SHOULD immediately remove the old Root CA
self-signed certificate from the trust anchor store.

In environments without such a directory service or repository, like
the Web PKI, recipients need a way to obtain the oldWithNew and
newWithOld certificates. The Root CA SHOULD include the subject
information access extension [RFC5280] with the accessMethod set to
id-ad-caRepository and the assessLocation set to the HTTP URL that
can be used to fetch a DER-encoded "certs-only" (simple PKI response)
message as specified in [RFC5272] in all of their self-signed
certificates. The Root CA SHOULD publish the "certs-only" message
with the oldWithNew certificate and the newWithOld certificate before
the subsequent Root CA self-signed certificate is released. The
"certs-only" message format allows certificates to be added and
removed from the bag of certificates over time, so the same HTTP URL
can be used throughout the lifetime of the Root CA.

In environments without such a directory service or repository,
recipients SHOULD keep both the old and replacement Root CA self-
signed certificates in the trust anchor store for some amount of time
to ensure that all end-entity certificates can be validated until
they expire. The recipient MAY keep the old Root CA self-signed
certificate until all of the certificates in the local cache that are
subordinate to it have expired.

Certification path construction is more complex when the trust anchor
store contains multiple self-signed certificates with the same
distinguished name. For this reason, the replacement Root CA self-
signed certificate SHOULD contain a different distinguished name than
the one it is replacing. One approach is to include a number as part
of the name that is incremented with each generation, such as
"Example CA", "Example CA G2", "Example CA G3", and so on.

Changing names from one generation to another can lead to confusion
when reviewing the history of a trust anchor store. To assist with
such review, a recipient MAY create an audit entry to capture the old
and replacement self-signed certificates.

The Root CA must securely back up the yet-to-be-deployed key pair.
If the Root CA stores the key pair in a hardware security module and
that module fails, the Root CA remains committed to the key pair that
is no longer available. This leaves the Root CA with no alternative
but to deploy a new self-signed certificate that contains a newly
generated key pair in the same manner as the initial self-signed
certificate, thus losing the benefits of the Hash Of Root Key
certificate extension altogether.

6. Security Considerations

The security considerations from [RFC5280] apply, especially the
discussion of self-issued certificates.

The Hash Of Root Key certificate extension facilitates the orderly
transition from one Root CA public key to the next by publishing the
hash value of the next-generation public key in the current
certificate. This allows a relying party to unambiguously recognize
the next-generation public key when it becomes available; however,
the full public key is not disclosed until the Root CA releases the
next-generation certificate. In this way, attackers cannot begin to
analyze the public key before the next-generation Root CA self-signed
certificate is released.

The Root CA needs to ensure that the public key in the next-
generation certificate is as strong or stronger than the key that it
is replacing. Of course, a significant advance in cryptoanalytic
capability can break the yet-to-be-deployed key pair. Such advances
are rare and difficult to predict. If such an advance occurs, the
Root CA remains committed to the now broken key. This leaves the
Root CA with no alternative but to deploy a new self-signed
certificate that contains a newly generated key pair, most likely
using a different signature algorithm, in the same manner as the
initial self-signed certificate, thus losing the benefits of the Hash
Of Root Key certificate extension altogether.

The Root CA needs to employ a hash function that is resistant to
preimage attacks [RFC4270]. A first-preimage attack against the hash
function would allow an attacker to find another input that results
in the hash value of the next-generation public key that was
published in the current certificate. For the attack to be
successful, the input would have to be a valid SubjectPublicKeyInfo
that contains a public key that corresponds to a private key known to

the attacker. A second-preimage attack becomes possible once the
Root CA releases the next-generation public key, which makes the
input to the hash function available to the attacker and everyone
else. Again, the attacker needs to find a valid SubjectPublicKeyInfo
that contains the public key that corresponds to a private key known
to the attacker. If the employed hash function is broken after the
Root CA publishes the self-signed certificate with the HashOfRootKey
certificate extension, an attacker would be able to trick the
recipient into installing the incorrect next-generation certificate
in the trust anchor store.

If an early release of the next-generation public key occurs and the
Root CA is concerned that attackers were given too much lead time to
analyze that public key, then the Root CA can transition to a freshly
generated key pair by rapidly performing two transitions. After the
first transition, the Root CA is using the key pair that suffered the
early release, and that transition causes the Root CA to generate the
subsequent Root key pair. The second transition occurs when the Root
CA is confident that the population of relying parties has completed
the first transition, and it takes the Root CA to the freshly
generated key pair. Of course, the second transition also causes the
Root CA to generate another key pair that is reserved for future use.
Queries for the CRLs associated with certificates that are
subordinate to the self-signed certificate can give some indication
of the number of relying parties that are still actively using the
self-signed certificates.

7. References

7.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <https://www.rfc-editor.org/info/rfc2119>.

   [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen,
              "Internet X.509 Public Key Infrastructure Certificate
              Management Protocol (CMP)", RFC 4210,
              DOI 10.17487/RFC4210, September 2005,
              <https://www.rfc-editor.org/info/rfc4210>.

   [RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic
              Hashes in Internet Protocols", RFC 4270,
              DOI 10.17487/RFC4270, November 2005,
              <https://www.rfc-editor.org/info/rfc4270>.

   [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS
              (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008,
              <https://www.rfc-editor.org/info/rfc5272>.

   [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
              Housley, R., and W. Polk, "Internet X.509 Public Key
              Infrastructure Certificate and Certificate Revocation List
              (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
              <https://www.rfc-editor.org/info/rfc5280>.

   [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the
              Public Key Infrastructure Using X.509 (PKIX)", RFC 5912,
              DOI 10.17487/RFC5912, June 2010,
              <https://www.rfc-editor.org/info/rfc5912>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [X680]   ITU-T, "Information technology -- Abstract Syntax Notation
              One (ASN.1): Specification of basic notation",
              ITU-T Recommendation X.680, August 2015.

   [X690]   ITU-T, "Information Technology -- ASN.1 encoding rules:
              Specification of Basic Encoding Rules (BER), Canonical
              Encoding Rules (CER) and Distinguished Encoding Rules
              (DER)", ITU-T Recommendation X.690, August 2015.

7.2. Informative References

   [SET]   MasterCard and VISA, "SET Secure Electronic Transaction
              Specification -- Book 2: Programmer's Guide, Version 1.0",
              May 1997.

Appendix A. ASN.1 Module

   The following ASN.1 module provides the complete definition of the
   HashOfRootKey certificate extension.

   <CODE BEGINS>

   HashedRootKeyCertExtn { 1 3 6 1 4 1 51483 0 1 }

   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   -- EXPORTS All

   IMPORTS

   HashAlgorithm
     FROM PKIX1-PSS-OAEP-Algorithms-2009 -- RFC 5912
           { 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) }

   EXTENSION
     FROM PKIX-CommonTypes-2009 -- RFC 5912
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-pkixCommon-02(57) } ;

   --
   -- Expand the certificate extensions list in RFC 5912
   --

   CertExtensions EXTENSION ::= {
      ext-HashOfRootKey, ... }

   --
   -- HashOfRootKey Certificate Extension
   --

   ext-HashOfRootKey EXTENSION ::= {   -- Only in Root CA certificates
      SYNTAX   HashedRootKey
      IDENTIFIED BY   id-ce-hashOfRootKey
      CRITICALITY   {FALSE} }

   HashedRootKey ::= SEQUENCE {
      hashAlg   HashAlgorithm,   -- Hash algorithm used
      hashValue   OCTET STRING }   -- Hash of DER-encoded
                                         --   SubjectPublicKeyInfo

```
   id-ce-hashOfRootKey OBJECT IDENTIFIER  ::=  { 1 3 6 1 4 1 51483 2 1 }

   END

   <CODE ENDS>
```

Acknowledgements

   The Secure Electronic Transaction (SET) [SET] specification published
   by MasterCard and VISA in 1997 includes a very similar certificate
   extension. The SET certificate extension has essentially the same
   semantics, but the syntax fairly different.

   CTIA - The Wireless Association - is developing a public key
   infrastructure that will make use of the certificate extension
   described in this document; the object identifiers used in the ASN.1
   module were assigned by CTIA.

   Many thanks to Stefan Santesson, Jim Schaad, Daniel Kahn Gillmor,
   Joel Halpern, Paul Hoffman, Rich Salz, and Ben Kaduk. Their reviews
   and comments greatly improved the document, especially the
   "Operational Considerations" and "Security Considerations" sections.

Author's Address

   Russ Housley
   Vigil Security
   516 Dranesville Road
   Herndon, VA 20170
   United States of America

   Email: housley@vigilsec.com