CoRE Working Group H. Mehrtens
Internet-Draft Universitaet Bremen TZI
Intended status: Informational February 14, 2014
Expires: August 18, 2014

Constrained Certificates for the Constrained Application Protocol (CoAP)
draft-mehrtens-core-ace-concert-00

Abstract

The present document defines a new kind of certificate suited for constrained environments. This new kind of certificate is intended to be used in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) in combination with Constrained Application Protocol (CoAP).

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [I-D.ietf-core-coap] uses DTLS [RFC6347] to establish a secure connection between distinct nodes in a sensor network. Like TLS, DTLS provides various ways to authenticate peers. For a certificate based authentication X.509 certificates or Raw Public Keys can be used [RFC5246], [I-D.ietf-tls-oob-pubkey].

X.509 certificates [RFC5280] were invented to fulfill the needs of the Public-Key infrastructure. This certificate format is very flexible and has lots of extensions, but that makes it also difficult to handle in constrained environments. In addition X.509 certificates are encoded using ASN.1 DER encoding, which needs complex parsers, compared to other formats.

An alternative for X.509 certificates is the use of raw public keys. They consist only of the public key and thus offer a very light-weight solution. But they provide no means for binding the public key to an entity. Moreover, many of the other features provided by X.509 are missing.

The intention of this draft is to define a new format for representing a certificate. This new kind of certificate aims to be a alternative to X.509 for constrained environments, while still being usable in the TLS and DTLS handshake like a X.509 certificate.

1.1. 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 RFC 2119 [RFC2119].

2. Use Cases and Problem Statement

When implementing DTLS with public key authentication on a Class 1 constrained node [I-D.ietf-lwig-terminology] using X.509 certificates tends to consume too much memory. The analysis done in [I-D.ietf-lwig-tls-minimal] on a constrained TLS implementation using X.509 certificates showed that the X.509 related code needed 2,776 Bytes and the ASN.1 parser needed an addition of 5,512 bytes.

Using raw public keys in DTLS already showed problems regarding the parsing and creation of ASN.1 structures needed for the ECDSA signature in the handshake. In a raw public key 27 bytes are used to describe the ECDSA capable key and to form the ASN.1 structure itself.

3. Design Requirements

The constrained certificates presented in this document are designed for the usage in TLS and DTLS with the Constrained Application Protocol (CoAP) and to replace X.509 in these environments.

The main approach is to create certificates better suited for the use in constrained environments. This is done by removing the elements from the basic structures of the X.509 format, which are not needed in this use case. In addition other elements are modified to include additions added by certificate extensions in X.509, which are needed in many use cases by TLS and DTLS.

While defining a new certificate format we used Binary Object Representation (CBOR) [RFC7049] to encode the data instead of ASN.1, because CBOR is better suited for constrained environments. The new format also makes use of the tagging mechanism of CBOR to describe elements.

4. Overview of the approach

The main differences in the design of Concerts compared to X.509 certificates are:

- Instead of using Object Identifiers as in X.509, tags from CBOR are used. Each of the object identifiers found in a small X.509 ECDSA certificate is between 3 and 9 Bytes long and is stored in an additional element whose definition needs additional space.

- Concerts will have a list of entity names the certificate is bound to and not just one subject distinguished name. There is no need for something like the subject alternative names extension, because all these names are defined in the list of subject names. This make the certificate itself smaller and it also makes the parsing of the certificate easier, because there is just one structure to parsed instead of two for X.509 certificates to get all subject distinguished names.

- One subject name in the list of subject names consists of only one value, such as the DNS name or the IP address. Many X.509 certificates used in the web also contain additional data in the subject distinguished name, describing the organization which operates the website, these information are not needed for the TLS or DTLS authentication.

- Instead of referencing the issuer certificate by its subject distinguished name, the issuer is referenced by a named information like defined in [RFC6920]. A SHA-256 hash over the public key of the issuer is computed and the first 8 to 32 bytes of that hash are stored in the certificate the issuer singed to identify issuer. This is similar to the use of the authority key identifier from from [RFC5280], Section 4.2.1.1. to identify the issuer certificate. This way Raw public keys [I-D.ietf-tls-oob-pubkey] can be used as certificate authorities.

5. Structure of the certificate

This new certificate type uses a similar ordering than X.509 does.

```
Certificate  ::=  Array  {
    tbsCertificate       TBSCertificate,
    signatureValue       Array  }

TBSCertificate  ::=  Array  {
    serialNumber         byte string,
    signature            integer,
    issuer               byte string,
    notBefore            dateTime,
    notAfter             dateTime,
    subjectList          SubjectNameList,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    extension            Map }

SubjectNameList  ::=  ARRAY OF SubjectName

SubjectName ::= CHOICE {
     IPv4               byte string,
     IPv6               byte string,
     dns                utf-8 string }

SubjectPublicKeyInfo  ::=  ARRAY {
     curve              integer,
     pubkey             byte string }
```

The Certificate element contains the complete certificate and will be tagged, to indicate a specific version of the concert certificate format.

The TBSCertificate element stores the data to be signed.

The serialNumber should be used by an issuer to store additional data like a serial number to do certificate revocation or to add some additional entropy to the certificate.

The signature element stores the signature algorithm this certificate was sign with. It must be the same value as used to tag the signatureValue element.

The issuer element must contain a named information defined in [RFC6920], which is a SHA-256 hash over the issuer’s public key or part of it. It is used to identify the issuer certificate. The element is tagged with the hash algorithm used to create the hash.

There are no secure hash algorithms needed, because a collision will not weaken the security, it will just prevent the issuer from being found.

The notBefore and notAfter element must meet the requirements for dateTime types.

The subject element is an array with the subject names this certificate is bound to. The subject names in the list must have a tag. There is a tag needed for an IPv4 address, IPv6 address and a DNS name. Using special representation for some identifiers makes it easier to compress the data and there is no code needed to convert an IP address from string into a binary representation.

The subjectPublicKeyInfo element contains an array with elements specific to the public key and is tagged with the public key type. In case of an ECDSA capable public keys a curve name and the public key itself are stored in this array.

The signatureValue element stores the signature of the tbsCertificate block made with the issuer private key. This element is tagged with the signature algorithm. That tag must match the tag of the signature element in the tbsCertificate element. The number of elements in the array depends on the needs of the signature algorithm used.

The ECDSA signature is not stored in an ASN.1 DER representation, but also encoded with CBOR. Both values of the ECDSA signature are stored in the array in binary representation with a fixed length depending on the signature type.

6. Example

Here is an example for such a certificate.

```
VERSION[
  [
    h'',                                      // no serialNumber
    SIG_ALGO,
    TRUNCATED_HASH(h'
      01 23 45 67 89 AB CD EF 01 23 45 67 89 AB CD
      EF'),
    1(1394751600),                            // 14/02/2014
    1(1402783200),                            // 15/05/2014
    [
      IPv4(h'C0000201'),                    // 192.0.2.1
      IPv6(h'20010DB80000000000000000000001'),
                                              // 2001:db8::1
      DNS("www.example.com")
    ],
    PUB_KEY([
      23,                                     // secp256r1
       h'04 CD 4E 80 9A DA BF 6B F7 BB 03 EF 9C 5C
         E0 0B C0 92 EB 94 14 04 5E C5 42 F2 57 99
         BD F8 42 88 96 24 1A 08 90 9A ED 2F C5 68
         BB 3C BC 48 20 78 08 7B D8 28 5C C9 ED 36
         65 A6 97 BA AB 62 D5 E7 95'
    ])
  ],
  SIG_ALGO([
    h'28 B1 51 1E 6F 03 10 12 8E 9A E0 3D 11 A2 F0
      AF BF 3D 1F EC 58 30 C3 FA 3E C4 F4 8B 75 40
      E8 17',
    h'D6 E4 0A 56 00 48 D7 BB F4 23 5B FC CB 5F 87
      52 0F 49 D8 F5 B2 85 8B EF B2 C1 27 17 2E F3
      A0 88'
  ])
]
```

7. References

7.1. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-07 (work in progress), February 2014.

[I-D.ietf-tls-oob-pubkey]
Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", draft-ietf-tls-oob-pubkey-11 (work in progress), January 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[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, May 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, April 2013.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

7.2. Informative References

[I-D.ietf-lwig-tls-minimal]
Kumar, S., Keoh, S., and H. Tschofenig, "A Hitchhiker’s Guide to the (Datagram) Transport Layer Security Protocol for Smart Objects and Constrained Node Networks",

draft-ietf-lwig-tls-minimal-00 (work in progress),
September 2013.

Author's Address

Hauke Mehrtens
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Email: hmehr@tzi.de